Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

NeoMiriX: an Artificial Intelligence System For Predicting Cancer Using miRNA Expression

Bishoy Tadros

bishoytadros2004@gmail.com

Badr University in Cairo

software

Keywords:

Posted Date: May 15th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9228225/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Abstract

Small RNA molecules control how genes work after they are copied, plus show abnormal patterns in nearly every kind of tumor. These signals stay intact in liquid samples like plasma, which sparked attention for medical testing - yet turning those readings into solid forecasts remains tough. Challenges pop up because studies differ widely in design while few methods combine DNA changes, RNA levels, and chemical tags across layers at once.

What if a tool could bridge the divide? NeoMiriX does exactly that by gathering miRNA expression alongside transcriptomic, genomic, and epigenomic layers from sources like TCGA, GEO, and CancerMIRNome. Instead of relying on one method, it combines several - Random Forest, SVM, and XGBoost analyze table-like patterns, while Transformers and Graph Neural Networks map how molecules interact. One piece ranks potential miRNAs, another explores biological pathways, yet another sorts patients by risk level.

What stands out is how well NeoMiriX performed across several cancer types - accuracy reached 0.96, while the ROC-AUC hit 0.97. Behind these numbers lies biological relevance: the miRNA patterns align closely with known cancer-related processes, particularly those guiding cell division and programmed cell death.

1. Introduction

Cancer still poses a major threat to people everywhere. Year after year, about ten million lose their lives, says the World Health Organization. By 2040, doctors may see more than thirty million newly detected cases globally. It ranks second among causes of death, trailing only heart conditions in scale. Some forms hit harder - lung, colon, breast, liver, and blood-related cancers lead in fatality rates. Unequal access shapes who suffers most under this weight. Surgeries have improved. New drugs strike precise targets. Immune systems now join the fight. Yet when found late, outcomes stay grim for many types. Big breakthroughs exist, but too often they arrive too late. The distance between treatment power and real-world survival remains wide.

Spotting cancer early changes everything when it comes to staying alive. Research after research shows people found with small, contained tumors live much longer - often double or even fivefold compared to those caught later (Siegel et al., 2023). Still, today’s go-to tools like cutting out tissue, examining cells under glass, or scanning the body come with problems - they hurt, cost too much, take time, and often miss tiny signs at first. A newer idea - testing blood instead - offers a gentler path forward, though it stumbles where precision fails and smart systems fall short in reading faint biological whispers correctly. Building smarter, number-crunching machines that catch cancer before symptoms appear isn’t just useful - it sits right at the top of what medicine must solve now.

MicroRNAs as Cancer Biomarkers

Tiny RNA bits called microRNAs - usually between 18 and 25 units long - tweak how genes work by latching onto specific message strands. Though they do not build proteins, these molecules fine-tune cellular activity after messages are copied. Back in 2002, Calin's team linked odd patterns in these RNAs to blood cancer, marking a turning point. From that moment on, researchers found them playing dual roles: sometimes driving tumors, other times blocking them. Across nearly every cancer type, their behavior shifts in ways tied directly to core disease traits. Out-of-sync levels help fuel unchecked growth, resistance to cell death, new blood vessel formation, plus spread to distant sites. These disruptions sit at the heart of what makes cancers aggressive. Evidence since then has only deepened that view.

What stands out is how miRNAs hold real promise for diagnosis because they last long in body fluids like blood, spit, and pee. Their presence shifts depending on which tissue or tumor type shows up, giving clues about what might be going wrong. Detection tools today reach deep enough to spot tiny amounts through modern tech such as gene scanners and chip arrays - accuracy improved thanks to earlier research (Mitchell et al., 2008). Collections including CancerMIRNome, miRBase, and TCGA now store mountains of miRNA activity records from many cancers, opening paths toward spotting trends hidden in vast numbers. Still, challenges pile up when handling these rich datasets - too few patient samples compared to variables measured, mismatches between study groups, varied biology among patients - pushing scientists to build smarter math models instead of relying only on classic stats.

The Multi Omics Approach in Cancer Studies

It turns out one kind of molecule alone cannot show everything about how cancer forms. So scientists now mix different kinds of biological data instead. Mutations in genes, shifts in gene copy numbers, chemical tags on DNA, activity levels of genes, and proteins present - all shape what a tumor looks like at the molecular level. Together they offer sharper insights compared to when studied separately (Subramanian et al., 2020). Projects like TCGA and ICGC gathered massive collections of such layered information from many patients and cancer types, making it possible to study them jointly. But combining these diverse measurements is hard - they vary wildly in size, units, meaning, and reliability - and turning them into stable, useful predictions remains tough work.

Machine learning and deep learning used in cancer research

Machine learning's role in cancer research grew fast during the last ten years, thanks to richer biological data and better algorithms. Not long ago, tools like Random Forest stood out for handling organized omics information well, giving clear signals about which features mattered most while still predicting accurately (Breiman, 2001). Around the same time, boosted models such as XGBoost picked up speed, showing similar strengths in interpretation and precision (Chen & Guestrin, 2016). Separately, Support Vector Machines carved a niche through smart use of kernels, managing complex gene-level sorting with notable success. Their knack for dealing with many variables made them fit naturally into genomics and RNA studies.

These days, systems built on deep learning push how far computers can go when spotting cancer signs. Starting out in language tasks, transformer-style networks now handle patterns across lengthy gene data and complex activity maps just fine (Vaswani et al., 2017; Theodoris et al., 2023). When it comes to cancer work, graph neural nets stand out because they map real biological links - like how genes, miRNAs, and proteins influence one another - instead of acting like each piece floats alone (Zitnik et al., 2018). Mix older statistical methods with these modern structures inside one workflow, yet still keep clear reasons behind predictions - not just raw accuracy.

Limits of Current Bioinformatics Tools

Even with advances, current bioinformatics systems for cancer forecasts and miRNA studies face key drawbacks affecting real-world use. While some tackle only one task at a time - like spotting gene shifts or linking markers to patient outcomes - they miss weaving steps together into one smooth flow from raw data to final medical insight (Huang et al., 2022). Often built for just one dataset, many lack the ability to blend information across sources like TCGA, GEO, or focused miRNA collections, weakening results when tested broadly. On top of that, explanation trails behind; most give answers without showing which genes, networks, or cellular functions shaped those conclusions - an obstacle for doctors trusting these outputs in care settings. Most current tools skip turning predictions into clear patient risk groups, leaving doctors without practical guidance despite advanced calculations. Without user-friendly design, many medical professionals find it hard to use these systems when coding skills are needed just to run them.

Explainable and Clinically Focused AI Systems

Putting AI to work in cancer care needs more than just correct guesses. Because rules, medical oversight, and ethics matter, any system giving diagnoses must show clear, understandable logic behind its results (Rudin, 2019; Tjoa & Guan, 2021). Tools like SHAP, visual attention maps, and pathway-based explanations help doctors see how predictions connect to biology, spot hidden influences, while gathering proof needed before using models on patients. Instead of black boxes, useful systems reveal their confidence levels, handle varied data types, even allow parts to change when science moves forward. When platforms miss these traits, progress slows down - trust lags, adoption stalls, risks grow. Without such features built in from the start, real-world use becomes shaky at best.

Neomirix scope and contributions

Facing gaps that still remain, our team developed NeoMiriX - a computing system powered by artificial intelligence focused on cancer signals from miRNAs. This tool supports spotting tumors early, finding new markers for disease, along with sorting patients by risk level in medical settings. Built as one complete flow, it handles diverse biological data types, blends different model styles for predictions, explains results in life-science terms, then delivers insights per person. Key advances here include these elements just listed - woven into a single working design

One way to look at many kinds of biological data is how NeoMiriX handles information from TCGA, GEO, or CancerMIRNome through uniform processing steps. Because it uses fixed methods for cleaning and aligning inputs, comparisons across different study groups become possible. Batch differences get adjusted in a structured way, while checks ensure reliability throughout. What matters is that noise fades without losing real signals - each step built to preserve truth.

A mix of old and new methods powers the system - traditional techniques like Random Forest, SVM, and XGBoost work alongside modern ones such as Transformers and Graph Neural Networks. These pieces fit together differently depending on the data's shape, pulling out patterns one method alone might miss. Together they handle complex biological signals across many cancer types, landing close to 96 percent correct guesses while staying sharp near 97 percent in distinguishing true hits from noise during tests.

Starting with raw data, the system sorts microRNAs based on how well they distinguish between tumor types. One after another, these markers get ranked by impact rather than abundance. What stands out isn’t noise - it's signal shaped by variation. Through this lens, patterns emerge that tie directly to patient outcomes. Not every molecule matters equally - some carry more weight simply because they appear consistently off-kilter. Step by step, the tool narrows in on those with real separation power. Behind each score is a trail of biological divergence. Clarity comes not from volume but from precision in difference.

Looking at pathways helps make sense of the results through known cancer-related signals. By tying predictions to these networks, the method reveals how things might be working inside cells.

Starting from raw predictions, NeoMiriX sorts patients into clear risk levels using a step-by-step method. Because outcomes need context, each category reflects practical medical decisions. Instead of vague scores, clinicians receive defined groups tied to real-world actions. This approach turns complex data into useful guidance over time. Since timing matters in care, the system helps spot who needs attention first.

Because models need clear reasoning, feature clues appear at every step. These hints help lab scientists and doctors understand results. Clear visibility into how choices emerge matters when AI moves into cancer care. Trust grows when everyone sees what drives a prediction. Seeing the why behind outputs keeps things grounded in real-world use.

What stands out is how each piece fits into NeoMiriX, shaping it into something that grows easily, works reliably, yet stays focused on real medical needs. Built for future miRNA cancer testing, its reach touches early diagnosis, tailored treatments, even choices guided by biological markers.

2. Literature Review

2.1 Machine Learning for Predicting Cancer

Years back, researchers began using labeled data to sort cancers by type or predict outcomes - this idea now sits deep in cancer science. Instead of one model alone, many blend their guesses; take how forests grow from single trees, each shaped differently yet part of a larger structure. That's what happens when separate tree-like models vote together after learning from repeated random slices of patient data. Even when thousands of gene signals compete for attention, these groupings rarely get fooled into memorizing noise. What stands out? Some inputs matter more than others - the method quietly ranks them, showing which molecules might be worth testing later in labs. One team found tiny RNA patterns correctly flagged diseased versus healthy cells over nine times out of ten, covering several tumors at once. Later efforts pushed further - not just spotting cancer, but naming subtypes or estimating time lived afterward. When clues hide within oceans of genetic blips, knowing where to look first saves months. Numbers rise, papers pile up, yet this tool keeps appearing - simple roots, lasting presence.

Out of all machine learning tools, Support Vector Machines stand out when sorting genetic data. These models work by finding the best possible dividing lines in complex data spaces, especially after applying special mathematical tricks called kernels. Back in 2000, Furey's team showed how well SVMs could tell apart different cancers using gene activity patterns measured from microarrays (Furey et al., 2000). Later on, scientists applied similar approaches to tiny RNA molecules, proving these models remain effective even when there are few samples but thousands of variables - a frequent challenge in medical genomics. The choice of kernel makes a big difference here. When cancer types create twisted, non-straightforward patterns in miRNA data, radial basis or polynomial versions tend to beat simpler linear ones, as Peng's group reported in 2009. Yet despite solid results, scaling up SVMs brings hurdles. They do not naturally give confidence scores, need increasingly heavy computation as data piles grow, and unlike tree-like models, their logic stays mostly hidden behind layers of math.

One reason XGBoost stands out in cancer prediction is how it builds decisions step by step using many small trees. While others rely on first-level gradients, this method uses sharper updates thanks to second-order math. Regularization built into the model keeps complexity under control, which helps avoid fitting too closely to noise. Over time, its speed and knack for dealing with incomplete entries made it a go-to for genetic and molecular data work. Studies led by Chicpo, then later Guo's team, showed it often wins head-to-head against older tree systems when sorting tumor types or forecasting outcomes. What also makes it useful - especially in medicine - is how it pairs with tools like SHAP to reveal which genes or markers influenced each call. Even though it's complex inside, researchers can still trace back why certain answers emerged.

2.2 Deep learning models applied to genomics and transcriptomics

One decade changed how computers handle genetic information because of new ways machines now learn. Because they see complex layers in data, smart systems do better than old-school methods at finding hidden links in genes. Pictures once trained these networks - then scientists tried them on DNA

instead. A team led by Alipanahi found such models spot gene switches directly from letters in code, beating hand-built references. Later work, like what Zeng's group did, used similar ideas to guess where proteins stick to RNA strands. When it comes to miRNAs, these molecules map landing spots, sort folded shapes, and flag signals tied to tumors. Filters inside act much like custom sensors tuned to meaningful chunks of sequence, as shown by Pan and Shen.

Starting off strong - transformers arrived via Vaswani's crew in 2017, using self-attention to shift how machines handle sequences. These models now lead the pack when it comes to reading biological strings of data, like genes and RNA. Instead of just following rules, they learn patterns hidden inside massive piles of text-like genetic code. One version called DNABERT trained on raw DNA letters, later helping spot key spots such as promoters or splicing zones. It didn’t need task-specific tweaks - worked right after training. Jump forward to Geneformer: fed with scRNA-seq readings from countless individual cells, it started forecasting what happens when genes get turned up or down. Hidden links within heart and brain diseases began showing up thanks to its internal mapping. Since attention lets distant pieces talk directly - no hand-holding needed - it handles tangled biology better than older tools. Long gaps between influential parts? Not a problem. Patterns form naturally during learning. Multi-layered omics jobs benefit because real-life traits often stem from scattered molecular whispers combining unexpectedly.

Picture a web where dots stand for genes, proteins, or tiny molecules. Lines link them when they interact. Scientists now treat cancer clues like such webs. These setups feed into special tools called Graph Neural Networks. Instead of reading data as lines or piles, these tools study connections. They pass hints from one dot to its neighbors, again and again. That shape - the pattern of links - holds meaning. Micro RNAs control genes in ways that shift with conditions. Their network shapes matter. A 2017 idea made it easier to move signals across such maps. One team used similar logic on medicine response forecasts. Another group focused on spotting key cancer-causing genes. Their model beat older styles based only on gene levels or raw code. When different types of lab measurements join the map - as traits tied to dots or lines - the tool adapts. It handles mixtures: DNA shifts, activity counts, molecule traces. Big cancer projects often blend many such layers. Such models fit well there.

2.3 Multi-Omics Integration Methods

Figuring out how to combine different types of molecular data has become a key hurdle in cancer research. That's because tumor growth, development, and treatment reactions involve many overlapping biological layers. One early way tried stacking gene, RNA, and epigenetic profiles together into one large set of inputs. But this method often fails - it treats all data types as equally useful even when they aren’t. Worse yet, sorting meaningful signals becomes nearly impossible when models already face too few samples compared to features (Huang et al., 2022).

Some smarter ways to combine data come from breaking down math grids or using network tricks. Instead of just stacking datasets, one method pulls out hidden patterns common across gene layers, showing groups that single views miss. A technique called Similarity Network Fusion builds separate

maps of patients for each test type, then slowly blends them together. This blending works like passing notes in class - each round sharpens the big picture by boosting agreements and quieting random blips. When tested on large cancer collections, these merged groupings tied much more clearly to how long people lived compared to looking at only one kind of measurement.

Lately, systems built on deep learning have taken things a step beyond earlier methods. Instead of treating data types separately, some models pull together information from multiple omics layers into one tight summary using a central compression point - this cleaned-up version then helps sort samples better later on (Ronen et al., 2019). Borrowing ideas originally meant for pairing images with text, certain setups now adjust how much each molecular type influences the outcome, making it clearer which ones matter most while also boosting results. Take MOMA: Cheerla and Gevaert introduced this approach in 2019 (Cheerla & Gevaert, 2019), combining DNA changes, gene activity switches, and RNA levels through smart weighting inside a neural network - it outperformed single-data-type strategies when identifying tumor subtypes and estimating patient lifespan across twelve different cancers studied in TCGA.

Even with progress, combining large-scale multi-omics data still poses tough technical hurdles. When merging datasets from separate studies - each using different sequencing tools, processing steps, batch layouts, or subject mixes - researchers must clean and adjust the data thoroughly; poor handling here might distort results by making models seem more accurate than they are. Running complex integration models on vast omics inputs demands heavy computing power, putting such methods out of reach for modest labs. Without shared standards to test how well various approaches work, judging one method against another stays unreliable.

2.4 Limits of Today's Bioinformatics Tools

Even though plenty of research exists on algorithms, turning machine learning and multi-omics methods into user-friendly, full-cycle tools for most cancer researchers hasn’t quite happened yet. Take cBioPortal for Cancer Genomics - it's among the top public platforms out there - offering strong visualizations and ways to explore TCGA and similar datasets, helping many gain easier access to vast cancer genomics information (Cerami et al., 2012; Gao et al., 2013). Still, at its core, cBioPortal works more like a data viewer than a space for building predictions: it doesn’t handle supervised machine learning, deep learning, or blending features across omics layers, sticking mostly to summaries, correlations, and survival plots. When people want to run predictive models using cBioPortal data, they have to pull the files out and shift everything to another system, leading to split processes where steps might differ slightly each time, raising risks of errors and inconsistent results.

One after another, different miRNA tools like miRSystem, miRTarBase, DIANA-miRPath, and miRNet offer help with finding targets, tracing pathways, or drawing networks - though none builds a full chain of predictions into one flow (Huang et al., 2011; Chou et al., 2018). Despite their usefulness, they run on separate tracks, rarely linking up with machine learning systems meant for sorting outcomes or estimating risks, so moving data means reworking it by hand every time. Because these programs don’t

speak the same language, analysts face repeated delays - a known hurdle that slows down work and makes large-scale studies harder to pull off.

Most platforms today struggle with a common flaw - no clear way to show how they reach decisions. Instead of revealing what specific genes, cellular processes, or connections influenced an outcome, they only display summary stats like accuracy or AUC scores. Doctors cannot easily trace why one result appears over another when such details stay hidden. In cancer care settings, trust grows only when experts can follow each step a system takes. When choices affect lives, opaque logic becomes harder to justify, especially if transparent options are available. Research by Rudin in 2019 made a strong case against treating models like closed boxes when openness is possible. Since then, more effort has poured into techniques such as SHAP values, visual cues for attention spans inside networks, and meaning-driven interpretations. Despite rapid progress in research papers, few of these advances appear within actual tools meant for daily medical use. Implementation trails far behind theory.

Sorting risks - turning smooth number predictions into clear patient groups that guide treatment choices - is something most current tools miss. When systems do offer such groupings, they often just list raw percentages, missing the real-world medical context, unclear margins, or practical guidance needed by cancer care teams. Building tools that blend strong prediction power with straightforward risk labels, links to how diseases behave in the body, and open reasoning stays unsolved; this gap shapes why NeoMiriX was built as shown here.

3. Datasets and Data Sources

3.1 Data Acquisition Strategy Overview

What makes NeoMiriX work comes from testing it on three big cancer data sources: The Cancer Genome Atlas, Gene Expression Omnibus, and CancerMIRNome. Chosen because they cover many tumor types, offer solid data, include varied patients, plus researchers trust them widely in cancer studies. Put together, these databases create a wide net - thousands of cases from different labs, technologies, covering many forms of cancer. This variety helps build predictions that hold up across groups, not just one narrow set. Before any modeling started, every dataset went through the same cleaning, adjusting, checking steps. Details sit in Section 4.

3.2 The Cancer Genome Atlas (TCGA)

Starting off with a massive team effort, the Cancer Genome Atlas emerged from cooperation between the National Cancer Institute and the National Human Genome Research Institute. This project mapped out the inner workings of human cancers, covering 33 kinds of tumors in detail. Instead of guessing, researchers relied on solid measurements drawn from more than 11,000 people treated for cancer. Each piece of data came from layers of biological signals collected in unison - genetic codes, activity levels, structural shifts - all woven together. Far beyond just one study, this collection stands as the richest,

best-documented cancer database we have today. Because of its depth, NeoMiriX used it heavily during early development, learning patterns and testing accuracy against real cases.

Right now, TCGA information comes through the GDC website - miRNA levels measured by small RNA sequencing show up here, along with linked gene activity (RNA-seq), genetic changes like mutations and copy shifts, plus DNA methylation details when they exist. Inside the miRNA group: 11,284 cancer specimens sit next to 727 healthy neighbor tissues, spread across 33 tumor forms such as LUAD, BRCA, LIHC, GBM, COAD/READ, OV, PAAD, and more you might not expect. Each sample tracks 2,588 mature miRNAs, labeled using miRBase v22, while expression strength appears as RPM - reads per million aligned to a miRNA. Patient records add depth: things like disease phase, cell structure rating, how long people lived, past therapies - all pulled in to help guide model training and time-to-event studies.

What makes TCGA matter for NeoMiriX isn't just one thing. Because it gathers many samples for each kind of cancer, predictions can be tested thoroughly - no guesswork needed. Data layers like genes and proteins come from identical patients, so combining them feels natural, not forced together after the fact. When information flows this smoothly, models learn patterns more honestly. Doctors' notes tied to each case help shape how risks get sorted later on. Outcomes start making sense because they reflect real patient histories, not abstract guesses. That link to actual medical records? It keeps the system's forecasts rooted in what has already happened.

3.3 Gene Expression Omnibus (GEO)

Housed at the National Center for Biotechnology Information, the Gene Expression Omnibus holds more public functional genomics data than any other archive on Earth. From labs across the globe come submissions involving microarrays, RNA-seq, single-cell analysis, and miRNA work - each added without central coordination. Unlike TCGA, where methods follow strict protocols, GEO collects results shaped by many different lab practices and tools. This variety brings complications when comparing datasets, especially due to shifts caused by technical differences rather than biology. Still, because it pulls from so many sources, rare cancers and less-studied patient groups often appear here before anywhere else.

From GEO, 47 datasets were pulled together for use in NeoMiriX, bringing in 8,320 samples tied to 18 forms of cancer. Chosen works focused on those offering raw or nearly unprocessed miRNA data along with patient details; left out were any with under 20 samples per category or missing key health records needed to assign classes. The tech behind these ranged from Affymetrix GeneChip arrays to Agilent microarrays and Illumina RNA sequencing - each requiring alignment through methods laid out later in Section 4.2. Included cancers filled gaps where TCGA offered little info, like nasopharyngeal carcinoma, thyroid papillary tumors, and diffuse large B-cell lymphoma, expanding what kinds of cases the system can sort accurately.

One key way the GEO part helps NeoMiriX is by expanding the range of lab settings used during model training and testing, so results aren't too tied to how TCGA data was made. Because of this broader exposure, models adapt better across different methods. Another benefit comes from certain GEO

collections that track patients over time, including their reactions to treatments - something TCGA lacks. With these records, tools can begin forecasting how therapies work or how illness moves forward, going past simple one-time diagnosis predictions common in microRNA research. This shift allows deeper insight into changing health states.

3.4 CancerMIRNome

CancerMIRNome isn't just another data dump - it zeroes in on how miRNAs behave across different human cancers. Built by pulling together results from lab work and computer models, it forms a tightly packed reference point rooted solely in cancer-linked microRNA patterns (Liu et al., 2021). While databases like TCGA and GEO cast wide nets, covering all kinds of genetic details, this one narrows the lens sharply on miRNA's role in tumors. Inside, you'll find more than raw numbers - each entry carries extra context: confirmed links between miRNAs and their targets, expression trends tied to specific cancers, biological roles teased out through analysis, along with ties linking disrupted miRNA activity to patient traits.

Inside NeoMiriX sits CancerMIRNome, holding data from 3,142 samples covering 20 cancers. It tracks 2,656 miRNAs, all tied to version 22 of miRBase. What sets it apart? Every miRNA links to proven gene targets, known pathways, and sometimes survival trends - verified through lab work. Because these connections exist, the system's scoring tool could rank miRNAs based on how meaningful they are in particular cancers. Pathway analysis used them too, matching unusual miRNA activity to major cancer-related signals like PI3K/AKT, MAPK/ERK, Wnt/β-catenin, and p53 networks. Clinical details built into the dataset also helped shape risk groupings, drawing lines between certain miRNA patterns and real patient results. That depth gave structure to predictions without guessing.

3.5 Dataset Characteristics Summary

A snapshot of key traits across the three datasets in NeoMiriX appears in Table 1 below. Each brings its own mix of samples and molecules. One begins with solid tumors, another leans into blood-based cancers. Dimensionality shifts noticeably between them. Cancer types span several major forms, though not evenly. Formats differ - some are matrices, others structured files. Their main role? Helping untangle biological patterns through analysis. Together, they form a varied but connected whole.

Table 1. Summary of datasets used in the development and evaluation of NeoMiriX.

Characteristic	TCGA	GEO (Curated)	CancerMIRNome
Total samples	12,011	8,320	3,142
Tumor samples	11,284	7,614	2,879
Normal/control samples	727	706	263
miRNA features	2,588	1,046–2,300*	2,656
Cancer types covered	33	18	20
Profiling platform	Illumina HiSeq (small RNA-seq)	Mixed (microarray & RNA-seq)	Mixed (curated multi-platform)
Expression unit	RPM (reads per million miRNA mapped)	Normalized intensity / CPM	Normalized expression score
Clinical annotation	Comprehensive (stage, grade, OS, DFS)	Variable (study-dependent)	Curated (outcome-associated)
Multi-omics layers	Yes (RNA-seq, DNA methylation, CNV, mutation)	Limited (miRNA-focused)	No (miRNA-centric)
Validated target interactions	No	No	Yes
Pathway annotations	Partial (via GSEA)	No	Yes (curated)
Primary role in NeoMiriX	Model training & benchmarking	Cross-platform generalization	Biomarker scoring & pathway analysis
Data access	GDC Portal (controlled/open access)	NCBI GEO (open access)	CancerMIRNome portal (open access)

*Feature count varies across GEO datasets due to platform heterogeneity; values reflect the range observed across the 47 curated datasets following per-platform annotation.

3.6 Data Integration and Harmonization Considerations

Putting together data from three different sources - each using separate tech platforms, scaling methods, leftover tagging versions - created real headaches for making things match up well enough to trust later predictions. When mixing omics data from multiple places one big snag shows up: what looks like biology might just be noise from lab steps, how deeply they sequenced, or chemical quirks tied to equipment (Leek et al., 2010). Here, we handled those shifts first by adjusting values inside each set to line up distributions, then used ComBat-seq (Zhang et al., 2020) across groups defined by source, while keeping disease types locked in so corrections didn’t wipe out actual patterns.

To line up the data from all three places, every ID got changed to match miRBase v22. The old ones were cleaned out, duplicates folded together when arm labels didn't agree across updates. Any feature missing in over 30 percent of samples in a single set was dropped before combining things. Gaps left behind filled quietly with k-nearest neighbor methods - kept totals whole but didn't warp patterns. After sorting, what remained held 22,473 samples, 1,847 microRNAs marked the same way throughout, plus health records allowing two kinds of guesses: tumor versus normal, or one of 33 cancer types.

4. Machine Learning Models and Training Methodology

4.1 Rationale for Model Selection

The selection of machine learning models for integration into the NeoMiriX platform was governed by three interconnected criteria: demonstrated empirical performance on high-dimensional omics classification tasks in the published literature; complementarity of inductive biases across the model ensemble; and compatibility with post-hoc interpretability frameworks required for biologically meaningful feature attribution. Classical supervised learning algorithms were prioritized for this component of the platform on the basis of their well-characterized behavior in settings where sample size is moderate relative to feature dimensionality—a condition that is characteristic of clinical miRNA datasets even at the scale of TCGA—and where model transparency is a prerequisite for scientific and clinical credibility. Three algorithms were selected to constitute the classical machine learning tier of NeoMiriX: Random Forest, Support Vector Machine with a radial basis function kernel, and XGBoost. Each algorithm embodies a fundamentally distinct learning strategy, and their combination within an ensemble voting framework provides robustness against the failure modes associated with any individual method. The theoretical basis and empirical motivation for each model are elaborated below, alongside a detailed account of the hyperparameter configurations adopted for this study.

4.2 Random Forest

4.2.1 Theoretical And Biological Foundations

Forest models, built using Breiman's 2001 framework, grow many decision trees. Each one learns from a fresh bootstrapped slice of the original data set. Predictions come together when votes are counted across all trees. What sets this method apart happens during splits inside each tree. Instead of checking every possible predictor, only a random group gets considered at each branch point. That twist keeps the trees from copying one another too closely. Less similarity among them means less swing in overall results. Because it uses both sampling tricks and scattered inputs, the whole system stays stable even with lots of variables. It holds up especially well when dealing with micro-RNA patterns. Those data types often pack hundreds - or nearly a thousand - features into studies where patient counts stay small. So mismatches between rows and columns do not throw it off easily.

When you look under the hood, Random Forest naturally weighs how much each feature helps split data more cleanly, averaging its effect everywhere it shows up across every tree - an approach that neatly

pinpoints key miRNAs tied to cancer traits without heavy computation. For NeoMiriX's system, this built-in ranking isn't the only clue but one piece among others feeding into which micro-RNAs get flagged as likely diagnostic markers.

4.2.2 Hyperparameter Settings Explained

A forest of 500 decision trees formed the core of the model, built using scikit-learn tools described by Pedregosa and team in 2011. Past this count, adding more trees barely improves accuracy, yet slows things down - shown through testing on similar sized data sets, as Oshiro's work highlights. Each tree grows up to twenty layers deep, just enough to detect intricate patterns among features without going too far. Too much depth leads to overfitting, especially when dealing with many microRNA traits and limited patient samples. When unchecked, such deep trees learn noise instead of signals, failing later on unseen cases, most notably in rare cancers where data is thin. Every time the model splits data, it uses a number of features equal to the square root of how many there are total - matches theory for sorting jobs, also what scikit-learn usually does. To balance things out when some cancer types show up way less than others, rarer ones get higher importance during training; this stops common cancers like BRCA and LUAD from drowning them out.

4.3 Support Vector Machine

4.3.1 Theoretical Basis and Biological Motivation

Out of the work by Cortes and Vapnik in 1995 came Support Vector Machines, later adapted for nonlinear cases using kernels that reshape how data points relate. Instead of treating every sample equally, these machines tune their attention where it matters most - on errors and close calls near the dividing line. What sets them apart lies partly in the hinge loss, which ignores distant correct predictions entirely. Control over model complexity enters through a single tuning knob, balancing simplicity against sensitivity. Because of the kernel trick, they reach into rich, stretched versions of the original space - all without actually computing each expanded coordinate. High dimensions stop being a roadblock when drawing curved lines between groups. Though built quietly behind math, their strength shows up clearly in messy biological datasets.

When looking at how miRNAs help sort cancers into groups, the tangled way they control traits makes straight-line methods fall short. These links involve layered networks, feedback after proteins form, and shifting access to targets - so simplicity fails. Instead of assuming clean splits in data, models must bend. Support vector machines with linear setups work fast but pretend biology fits neat borders, which it does not. Gene silence shaped by miRNAs spreads across many dimensions, defying flat separation. A better path uses the radial basis function kernel - it measures likeness by shrinking scores as points drift apart in space. This shape-shifting line adapts where rigid ones cannot, matching patterns more closely. Past tests show it outperforms simpler versions when grouping tumors using miRNA signals (Peng et al., 2009; Guo et al., 2021).

4.3.2 Hyperparameter Settings Explained

A model based on support vector machines used scikit-learn's SVC tool, picking the radial basis function for shaping boundaries. Instead of default settings, it leaned into tighter constraints by choosing C equal to 10 - this keeps mistakes during learning in check while still allowing some flexibility. Too high a C might lock onto every training detail, creating fragile patterns; here, balance mattered more than perfection. Through repeated testing with five levels of internal checks, this setting outperformed others like 0.01 or 100 when measuring average AUC scores across different tumor groups. For how far each data point influences others, gamma took a practical route called 'scale,' built from feature count and data spread. That choice adapted automatically instead of fixing one rigid number. Scaling adjusts automatically because miRNA data has many dimensions, so the model avoids clustering too tightly around individual samples or blurring distinctions among similar cancer types. To get reliable likelihood scores for classes, probabilities were fine tuned using Platt scaling, making them fit better within systems that combine predictions and sort patients by risk level.

4.4 XGBoost

4.4.1 Theoretical Basis and Biological Motivation

Start strong with XGBoost - it came from work by Chen and Guestrin back in 2016. Not like others, it stacks small decision trees one after another, slowly improving predictions. Each new tree? It targets the direction where error drops fastest, guided by a chosen loss measure. Random Forest does things differently: many trees grow at once using resampled data, then average results. But here, every next step fixes what previous ones missed too far off track. More trees mean less bias, though how fast depends on the size of each update. What sets this apart begins with smarter math - it uses curvature clues from second derivatives. That leads to sharper moves toward better answers than first guesses alone allow. On top, penalties are placed directly on leaf values and splits, both sparsity-inducing and smoothness-enforcing kinds. Also built in: picking only some features per split, much like forest methods do. This mix lowers overlap across trees, helping avoid overfitting while lifting real-world performance.

Because it handles complex patterns well, gradient boosting fits neatly into tasks sorting many cancer types using miRNA data. Step by step, through repeated adjustments, XGBoost picks up tiny differences among cancers that look alike in their overall miRNA activity yet shift slightly in key markers - spots where simpler models often stumble. Built right in, its design allows precise Shapley value calculations tied to every forecast, linking predictions back to specific miRNAs with clarity. These clear links feed straight into NeoMiriX, strengthening how it explains results and ranks potential biological signals.

4.4.2 Hyperparameter Settings Explained

To build the model, we used the xgboost package in Python with specific settings. After every step, performance was checked on held-out data using log-loss; if it didn’t improve for 50 steps straight, training stopped - so it ended where results were best, not just after a preset number. One thousand

possible steps were allowed, though usually fewer were needed due to early halting. Each new tree had its impact reduced by multiplying by 0.05 before being added, slowing down learning but often helping later accuracy. Because adjustments per tree stayed small, many trees became necessary, yet the whole system adapted more smoothly to patterns instead of noise. Individual trees could split nodes up to six levels deep, limiting how intricate they got and reducing chances of capturing rare combinations only present in one group of samples. Complex interactions beyond that level weren't fully explored, since such details tend not to repeat well in unseen cases. Lots of randomness came into play every time a new boost happened, with 80 percent of data points and inputs picked by chance - kinda like how trees grow differently in a forest when fed random subsets. A method called softmax handled the guessing game among 33 types of cancer, pushing predictions toward better accuracy using cross-entropy rules. Uneven numbers across cancer groups? That got balanced out inside the scoring system so rarer forms didn’t get ignored.

4.5 Training Methods and Testing Approach

Each of the three classic machine learning setups went through training and testing under one consistent setup meant to measure results fairly. Not quite randomly, the full collection of 22,473 cases got split into pieces: one part for training, another for checking progress, and a last piece kept separate for final evaluation. To keep things balanced, this division made sure every cancer category and normal versus tumor group stayed proportionally present in each section. Most of the data, about seventy percent, fed into building the models; fifteen percent helped fine-tune settings along the way. The rest - also fifteen percent - sat untouched until the very end, used only once predictions were ready to be judged. Within the main training chunk, researchers ran five rounds of structured checks where different configuration choices could be tested. Steps like scaling values, adjusting for lab-specific noise, and filling in absent numbers were learned strictly from just those training segments. Then they carried forward without retraining, applying them unchanged to check points during assessment. Only after everything else was locked in did the final scoring happen - using that isolated holdback slice never seen by any model before.

Out of several options, the method picked for NeoMiriX’s main prediction uses a mix of outputs from Random Forest, SVM, and XGBoost. Instead of picking one winner per model, their likelihood scores for each cancer type get averaged together. Each model counts just as much as the others, after tests showed none clearly outperformed the rest when checked repeatedly. Someone tried giving XGBoost more influence, yet that didn’t help much overall. Sticking with equal shares made things simpler without losing accuracy. For any given case, whatever diagnosis gets the strongest average vote becomes the system’s call. That full set of odds then moves forward into the next step, where patients are grouped by risk level. Every detail about how these three models were trained can be found in Table 2.

Table 2. Hyperparameter configurations for the classical machine learning models implemented in NeoMiriX.

Parameter	Random Forest	SVM (RBF)	XGBoost
Primary estimators	500 trees	—	1,000 boosting rounds
Maximum depth	20	—	6
Learning rate	—	—	0.05
Kernel / split criterion	Gini impurity	RBF	Softmax (multi-class)
Regularization	Max depth, min samples	C = 10	L1/L2 + subsampling
Kernel coefficient (gamma)	—	Scale	—
Feature subsampling	√(n_features) per split	—	0.8 per round
Sample subsampling	Bootstrap (100%)	—	0.8 per round
Class imbalance handling	Inverse frequency weighting	Inverse frequency weighting	Inverse frequency weighting
Probability calibration	Enabled (Platt scaling)	Enabled (Platt scaling)	Native (softmax)
Early stopping	—	—	Patience = 50 rounds
Implementation framework	scikit-learn v1.3	scikit-learn v1.3	xgboost v2.0
Optimization criterion	Cross-validation AUC	Cross-validation AUC	Validation log-loss

Not every tool works the same way, yet together they cover more ground. Instead of relying on just one approach, using all three spreads out risk. One stabilizes feature rankings even when inputs overlap. Another draws clean dividing lines in complex data settings. The third sharpens predictions step by step while showing how each piece influences results. Each sees things slightly different. Outcomes gain strength because weaknesses in one get balanced by strengths in another. Confidence rises when separate methods point to similar conclusions. This matters deeply in cancer research where trust in why a result appears can matter as much as the result itself.

5. NeoMiriX Platform Software Structure

5.1 Architecture and How It Was Designed

Built like building blocks stacked one on another, NeoMiriX splits tasks such as reading data, cleaning it, training models, making forecasts, explaining results, and sharing reports into separate pieces. Each piece works alone yet talks to its neighbors using clear rules for exchange. Because of this setup - shaped by ideas about keeping jobs distinct and focused - it does two things really well when used in medical data work. When new methods emerge or science moves forward, any part can change without

touching the rest. Testing also becomes easier since checks happen piece by piece, helping ensure consistency, traceability, and trust when applied to real-world health studies or future patient care tools.

Python 3.10 runs the core of this setup, using tools like NumPy and pandas for number work, while scikit-learn and XGBoost handle traditional machine learning - deep learning comes alive through PyTorch plus its graph extension. Instead of relying on web tech, it uses PySide6 to build a desktop front-end that works just the same whether you're on Windows, Mac, or Linux. Behind the scenes, tasks drag out over time without freezing the screen thanks to asyncio managing threads quietly in the background. When things get heavy - like training models or shifting big datasets - the app stays snappy because work happens off the main thread. A diagram labeled Fig. 1 shows how everything connects at a high level, breaking the whole thing into six major parts stacked by purpose. One piece handles raw data flow before feeding results downstream; another takes charge once modeling begins. Predictions come from a dedicated engine tuned for speed and consistency when serving outputs. Reports form automatically after analysis completes, shaped by rules tucked inside a separate module. External systems reach in via an API layer meant for integration beyond the local machine. Finally, deployment and scaling needs sit on their own foundation so updates and distribution stay manageable across machines.

5.2 Data Processing Pipeline

Instead of starting from zero each time, the system builds on earlier steps, shaping messy biological data from various databases into something tidy and ready for analysis. Each stage follows a set pattern, built around a template called BaseProcessor that checks inputs, changes formats when needed, then reverses those shifts if required. Though the flow moves forward only, its parts snap together like blocks, arranged in sequence depending on what the task demands right then. Even after training ends, the setup remembers how it was shaped before, so later data passes through the exact same path without doing everything again.

Starting off, data moves into the system using special tools made for each source - one for TCGA, one for GEO, one for CancerMIRNome. These tools manage login steps, pull down files, interpret file formats, then reshape everything to fit a common structure. As raw gene activity tables arrive, they become pandas DataFrames, their columns renamed uniformly while matching microRNA labels to miRBase version 22 right away. An automated checker examines every dataset for shape and meaning, catching issues like unusually small sequencing depth, too many blank entries, or odd expression patterns prior to any cleanup steps.

One step at a time, the data gets cleaned and aligned through standard steps. Noticing uneven coverage? Quantile normalization smooths out those bumps caused by varying sequence counts and sample makeup. After that, ComBat-seq adjusts for lab-to-lab shifts - each dataset treated as its own batch while keeping disease types untouched in the math. When values go missing, they’re filled in using nearby examples, trained only on the initial group so nothing sneaks into later sets. Then come the keep-or-toss choices: first dump any signal too flat to matter, then pick what moves most with diagnosis using

F-scores from ANOVA-style checks. What remains becomes the small set of microRNAs used when teaching the final predictor.

A single ValidationPipeline class holds every step of preprocessing together. Inside it, each processor runs one after another in order. A record of what happens gets saved along the way. This log helps show exactly how things were done. Training data goes through a fit_transform process. Data kept aside - plus anything used later during predictions - only passes through transform. Parameters never borrow information from samples meant for testing. What shapes the model must come purely from training examples. Later inputs stay untouched by those rules once they are set.

5.3 Model Training Pipeline

One step at a time, the system sets up how each prediction tool behaves during learning and testing across two layers - traditional algorithms alongside neural networks. Sitting in charge, a component called ModelPersistenceManager keeps track of every design option, saves progress with clear labels, stores results safely after training wraps up. Each saved piece gets tucked away neatly so it can wake up later exactly as it was, ready to make decisions again when pulled back into action. This bridge between storage and reuse means past work never needs repeating.

A TrainingPipeline class begins the process, taking a model config plus a cleaned feature set. Splitting happens next - data divides carefully into train and validate chunks, keeping groups balanced. This setup feeds a cycle that runs training for whatever model type was chosen. Older ML methods hand off work directly to tools like scikit-learn or xgboost instead. Their settings arrive as labeled maps of values, saved at once with trained weights so nothing gets lost later. That way, every run can be rebuilt exactly when needed. Neural networks follow another route - one built on PyTorch from the start. Gradients form step by step while updates move through layers under control of something called DeepTrainer. It handles timing for learning shifts, when to stop early, how steps unfold - all behind one steady front. Same structure works whether facing Transformers or graph-based nets.

Fine-tuning settings happens using built-in cross-validation checks inside the training split, keeping the validation data untouched. As learning unfolds, each step gets captured automatically through NeoMiriXLogger, storing loss patterns and scores at every stage across log files and visual output panels. Model outputs - including weights, tuning choices, processing steps, selected features, and results - are saved systematically into labeled folders tied to unique version tags, making past runs traceable and redeployable when needed.

5.4 Prediction Engine

Inside runs a system built to take fresh patient details. This part grabs incoming information, slips it through ready-made cleanup steps. One piece fires up several trained models at once. Their separate answers get pulled together, shaped into clear health risk levels doctors can grasp. Code ties everything inside something called PredictionPipeline. Think of it like one box doing many jobs when asked. Drop gene activity numbers in different forms - plain spreadsheets, tab files, HDF5 chunks, even GEO soft

entries - it handles them all. Out comes an organized result: odds for each category, grouped danger level, key markers ranked, active biological paths listed. Not magic. Just logic wired tight.

When making predictions, each model in the group handles the processed data separately - Random Forest, SVM, XGBoost, Transformer, and Graph Neural Network. Because they work alone at first, their outputs differ slightly. Once done, their predicted probabilities combine using weighted averaging, where some models can influence more than others. This combined result moves into the RiskStratification step. There, a tested rule system checks two things: the highest probability and how much disagreement exists across models. Depending on those, every case gets sorted into one of four levels - low, intermediate, high, or critical risk. If the models strongly disagree, it suggests confusion due to unclear patterns in the molecules. That mismatch raises a signal. A note attaches to the outcome, suggesting extra care when reading it alongside patient details. While not certain, such cases need closer human review.

Inside the system, explanations come out by default during prediction instead of being added later. For every forecast made, SHAP TreeExplainer handles XGBoost while KernelExplainer works on Random Forest, calculating individual feature impacts. At the same time, outputs include attention weights pulled from the last encoder stage of the Transformer model along with relevance scores from nodes in the GNN. What builds up across these different models is a unified ranking of biological markers. That combined signal flows straight into generating the final output report without delays or extra steps.

5.5 Report Generation Subsystem

From raw data shaped by the prediction module, readable documents emerge - crafted for scientists and medical staff alike. Housed inside a component called ReportGenerator, formatting begins once inputs arrive: one analysis result plus a layout chosen by the user. Filling each section happens next - findings slotted in place alongside charts, explanations, visual elements. What comes out takes form as PDFs, web pages, or organized JSON files, ready for review without extra steps. Built this way, it bridges machine output and professional needs quietly, without fuss.

One part shows what cancer type the system guessed, how serious it seems, also how sure the model feels about that choice. After that comes a look at key biological markers, listing most influential miRNAs based on combined impact scores, their activity levels compared to typical ranges seen during development. Instead of just listing molecules, this view connects them to known disease-related pathways, showing which signals appear disrupted through calculated relevance strength. Each result ties back to exact tools used, recording software versions, data adjustments made, sources of information fed into the process, so others can follow along later.

Inside the report, visuals like bars showing feature importance, color grids for pathway activity, along with dials that mark risk levels - pop up automatically through code built on matplotlib. When the system allows live interactions, these shift into dynamic versions via Plotly instead. Each image forms without

human help during the report build, tucked right in as part of the flow. Because it runs on its own, every output matches the last, staying accurate even when making many at once.

5.6 API Design

From behind the scenes, NeoMiriX opens up its analysis tools via a well-organized API. This setup allows script-based connections to fit within current bioinformatics processes, while also supporting hands-on work through the visual side of the platform. Grouped into four areas - data, models, predict, and report - the system offers distinct functions for different tasks. Every section comes with specific entry points, each explained in clear documentation. Inputs and outputs follow strict templates built with Python dataclasses. Validation happens live during execution, powered by the pydantic framework. Clarity emerges when structure meets real-time checks across every call made.

Inside the data area you will find tools to bring in datasets, build steps for cleaning data, while pulling ready-to-use feature sets from an embedded SQLite store. Tools under models let users train systems, check their performance, save results, using setup guides that define structure, tuning numbers, plus links to training material. When it comes to predicting, one main function takes unprocessed gene readings then delivers complete outcome records, along with finer options to pull single model answers or contribution measures on demand. Reporting tasks rely on utilities that pick layouts, generate documents, ship them out in accepted file types.

When async support exists, heavy computational functions offer coroutine versions returning awaitables - so they fit smoothly into async apps without blocking. Instead of running directly in code, NeoMiriX can work as a backend service via a REST interface built on FastAPI. This setup takes regular Python inputs and turns them into JSON, sending responses back the same way. Every endpoint shape comes documented automatically, thanks to type hints defined with pydantic - no extra steps needed. The system checks user access using validated JWT tokens, making sure only authorized people reach sensitive parts. Designed this way, it fits institutions needing strict data handling across many users.

5.7 Scaling and deploying systems

NeoMiriX handles growth by adjusting design across different system layers, shaped around how it might run - on one person's computer or big shared systems in the cloud. Instead of loading everything at once, huge data tables sit in HDF5 files using h5py, so pieces can be pulled when needed without overwhelming memory. As each step moves forward, only active chunks stay loaded; what isn’t used now slips away quietly behind the scenes.

Running several cross-validation tasks at once happens via ProcessPoolExecutor from Python’s built-in concurrency tools - this spreads the load across available CPU cores. Deep learning fits take advantage of PyTorch’s multi-GPU abilities when needed, switching to DistributedDataParallel if a model grows too large for one GPU. When predictions come in fast, the system handles them safely using lock mechanisms inside ModelPersistenceManager, so multiple users can get results at the same time without interference. This keeps the web interface responsive even under heavy loads.

A Docker image comes ready-made, wrapping up every needed tool so the setup works the same everywhere it's launched. For bigger setups, there's a Kubernetes plan included, helping spread predictions across many machines when demand grows. During lengthy runs, especially with big groups of patients, memory gets checked often while cleanup routines keep things from piling up. From one sample on a desktop to massive batches on clusters - everything runs unchanged under the hood.

6. Database Architecture

6.1 Overview And Design Rationale

What holds data in NeoMiriX isn't one size fits all. Instead, different kinds live where they work best. Structured information like user details or activity logs settles into PostgreSQL - clear rows and columns doing what they do well. Meanwhile, messy, shifting biology files, say gene readings or analysis outputs, find room in MongoDB, flexible enough to bend without breaking. Speed matters too. Things needed fast, gone soon - like active sessions or fresh results - live in Redis, always ready in memory. These pieces stay separate but talk through one clear interface. That middle layer handles how things connect, ask questions, send back answers, so the rest of the software does not have to care which part stores what. Three systems. One way to reach them.

6.2 PostgreSQL Relational Schema

Later on came the choice of PostgreSQL - solid when it comes to handling intricate queries across many joined pieces. Its grip on ACID principles made a difference early. Connection pools, replication setups, those fit well too. Five main tables form the backbone now: Users stand first, then Datasets follow close behind. After them step Models, feeding into Predictions next. Last in line sits Reports, tying bits together quietly.

Authentication details, roles, and user account data live inside the Users table for everyone signed up on the platform. A unique ID built as a UUID marks each row - assigned when an account first appears - not counting on regular numbers that follow one after another. Usernames sit next to encrypted passwords, along with official email addresses tied to institutions. One field spells out whether someone is a researcher, clinician, or admin - no mixing between these types. Timestamps track both signup time and most recent sign-in moment, giving visibility into activity patterns. An extra column holds true-or-false values showing if an account remains enabled. Because every record carries a UUID instead of numbered tags, outsiders cannot guess how many people have joined. Even when several servers add rows at once, duplicates won't happen thanks to the randomness baked into those IDs.

Holding every dataset brought into the system, the Datasets table logs identifiers along with where they came from - like TCGA, GEO, or CancerMIRNome. Each entry shows which cancer type - or types - is included, how many samples exist, plus how many features were captured during processing. The version of the cleanup method used gets noted at the moment it enters. Time of arrival is stamped automatically. Ownership links back to a user through a related ID field. When files arrive, their raw form

is checked; an MD5 fingerprint goes into a special column. Later lookups can compare that value again, spotting accidental changes by checking consistency over time. That hash acts like a snapshot taken right at intake.

Inside the system, a table called Models keeps track of every trained model along with its setup details. One entry holds things like the model's ID, structure type, settings saved in JSONB format, data used for training, test results, when it was trained, where the file lives, and which version it is. PostgreSQL's built-in JSONB feature allows different models to store unique configurations freely inside one shared space. Even so, individual setting values can still be checked directly by the database engine - no need to unpack everything through external scripts.

Holding results from every analysis run, the Predictions table logs details like the unique ID for each forecast alongside the person's ID provided when starting things off. A specific mix of models leaves its mark here too - their version tags get saved along with what kind of cancer was flagged in the outcome. Risk level shows up as one piece; another is how sure the system feels about that top guess. Differences in agreement across models slip into view through a spread measure tucked beside timestamps marking exactly when everything came together. Tied back to whoever asked for it, each entry finds its place linked to a user profile behind the scenes. Over in the Reports table, documents pulled from these forecasts take shape slowly in storage rows. Each ties directly to just one earlier prediction using a reference pointer built on shared IDs. Format choices live inside entries, plus which layout draft shaped the final look. When the paperwork finished forming gets noted down precisely, while where it landed digitally hides within path strings pointing at real files sitting ready somewhere else.

Foreign keys lock referential integrity tight across all five tables, while multi-table writes wrap inside explicit transactions - keeping changes atomic. When a user account vanishes, linked predictions and reports follow it down, yet datasets and models stay untouched, since they might serve others too. Deletion rules lean cautious, avoiding chain reactions where sharing happens.

6.2.1 Indexing Strategy

Every so often, speed matters most when pulling data fast. Primary keys across all five tables get B-tree treatment just because it helps things move smoothly. Instead of single columns, some lookups team up - like user_id with prediction_timestamp in Predictions. Another combo there ties cancer_type and risk_tier together for group-style searches. These pairs handle requests that sort personal forecasts by time or gather cases by medical profile. Meanwhile, behind the scenes, only live models need attention during launch. So the Models table uses a trimmed-down index, skipping anything inactive. That slice speeds up boot-time checks without touching outdated entries. A single entry per checksum appears in the Datasets table, thanks to a strict index that blocks duplicates even when files arrive labeled differently. Inside Models, settings live as structured snippets, searchable fast because GIN marks every piece of those JSONB fields. No sweeping through everything needed each time someone asks what

configurations exist - or which ones hold certain keys. Same data slipping in twice gets stopped cold where it lands - the database enforces cleanliness by design.

6.2.2 Connection Pooling

Starting up a fresh link to PostgreSQL takes noticeable time. Because of that, linking each app request straight to the database falls apart under heavy user loads. Instead, NeoMiriX uses PgBouncer - a lean tool tucked beside the main server - to reuse existing links efficiently. Running in transaction-focused mode, it keeps a live connection open just long enough to finish one operation. Once done - commit or abort - it sends the slot back into rotation instantly. Most actions taken by NeoMiriX involve brief updates or quick lookups using indexes. That rhythm fits perfectly with how this system handles sharing. Though limited, its design matches real usage closely. When the system needs more database links, it can open up to 100 on the server side. Clients may connect up to a thousand times at once. If getting a link takes longer than five seconds, an alert appears in the API part. Instead of waiting forever, the setup flags that resources ran out. Over time, small checks test active links so broken ones get removed early. This cleanup happens before problems grow during busy periods. Old connections vanish quietly, making room for fresh ones. Because of this cycle, performance stays steady even when demand climbs

6.3 MongoDB Document Collections

Out here, MongoDB holds biological data that’s just too complex or unpredictable for regular database setups. Expression matrices go into one pile. Feature importance lands in another. Shapes differ so much between entries that rigid tables would get messy fast. These two groups stay separate on purpose.

Inside the expression_matrices collection sits cleaned-up miRNA data from each patient. One entry matches one sample, holding its unique ID along with a link to the main Datasets table in PostgreSQL. You will find a tag showing the cancer type, plus an ID for where the data came from. It also includes which version of the processing method was used. Expression levels appear as a lean array built only for tested miRNAs, using miRBase v22 codes as labels. That slim format cuts down space when older machines skip many markers - a usual case among varied GEO inputs gathered here. Tucked within each record is extra detail: what machine ran it, how the genetic material was prepared, and how numbers were adjusted afterward. This background stays visible so later checks on quality or comparisons between groups stay possible.

Inside the feature_importance set lives data about how each model and each prediction uses its inputs, pulled out by explanation tools built into the forecasting system. One entry ties together the ID of a forecast, the design of the model behind it, the way importance was measured - be that SHAP TreeExplainer, KernelExplainer, Transformer attention, or GNN node scoring - and a list of miRNA IDs sorted by impact strength. That same record also rolls up those fine-grained scores, linking each miRNA to broader cancer-related signal routes through known gene targets. Structure shifts depending on what kind of model made the call: Transformer attention shapes results like layered grids from multiple heads, whereas SHAP spits out straight rows of numbers. This mismatch in form doesn’t break anything

because MongoDB handles mixed layouts smoothly inside single entries, skipping rigid tables or empty slots entirely.

6.3.1 Indexing Strategy

Indexes in MongoDB match how apps usually pull data from every collection. Because model training often grabs all samples of one cancer type inside a dataset, the expression_matrices table uses a combined setup: dataset_id first, then cancer_type. Fetching just one sample's processed gene data happens through a solo index on sample_id - common when running predictions. For finding importance scores tied to a certain forecast, prediction_id gets its own index in feature_importance. When ranking key features across many models for a cancer kind, sorting relies on two fields together: model_architecture and cancer_type. Background building keeps reads and writes moving while these structures form on big datasets.

6.4 Redis Caching Architecture

Holding data temporarily, Redis speeds up access for information that shapes how quickly users feel the system responds. What lives inside includes saved guesses from models along with active visitor sessions. Fast lookups happen because everything stays in memory, cutting delays where timing matters most. Instead of waiting on slower storage, frequently used pieces come straight from RAM. One part tracks what a person is doing during their visit. Another keeps recent results ready so they do not need recalculating. Performance gains show up right away when requests hit these stored points. Speed comes from design - data sits close to processing, reducing wait times dramatically. No disk bottlenecks slow down reads meant to be nearly instant. Both functions rely on quick writes and even quicker fetches behind the scenes. User actions stay fluid since past steps are remembered efficiently. Predictions reappear fast if asked again moments later. This layer handles rapid changes without dropping pace. Short-lived but critical bits pass through here constantly. Everything works together by keeping only what must be swift inside this space. Latency drops because distance between request and answer shrinks completely. Memory acts as both store and delivery path at once. Temporary does not mean less important - it means timed perfectly. Access patterns favor speed above long-term retention here. Immediate response needs shape how it is built and used.

Sometimes the very same gene patterns show up more than once - maybe a scientist runs the same test again using another form, or a hospital system asks for fresh results on an existing file. Running the whole analysis each time takes seconds, especially with detailed explanations and pathway ratings included. Getting back a stored answer instead cuts wait time down to less than a hundredth of a second. What decides whether something matches? A digital fingerprint made by combining cleaned-up data numbers with the current model setup code, locked in via SHA-256. That way, if the system upgrades its brain, old answers get tossed out naturally - even if the input looks familiar. Each saved response gets packed into a lightweight format, lives for one day, then disappears. This keeps memory usage steady while still catching plenty of repeat cases.

Inside Redis, ongoing user sessions keep track of JWT details, when users last acted, along with short-lived app data. Each record ties to a token ID, vanishing after a set time - eight hours for researchers, but only two for clinicians due to tighter rules. Updates on training tasks flow via Redis messaging, sent from backend workers straight to the main server. These signals move onward to open client connections using event streams, showing current status visually. The interface stays updated instantly, no repeated checks needed.

One main Redis node runs alongside two copies that handle reading tasks. Should the main node go down, Redis Sentinel steps in without delay to shift control. Data flow from primary to replicas stays under constant watch for delays. For checks needing up-to-the-second accuracy, like confirming user sessions, requests hit the original source only. When timing matters less, say pulling stored forecasts, reads spread out among backups. This split keeps pressure off the central point. Balance shifts naturally based on what each query demands.

6.5 Unified Data Access Layer

Under one roof sits the data access layer, wrapping three kinds of storage using separate repositories. UserRepository handles users while DatasetRepository manages datasets - both tied to PostgreSQL along with ModelRepository, PredictionRepository, and ReportRepository. On another path entirely, ExpressionMatrixRepository pulls expression matrices from MongoDB, just like FeatureImportanceRepository grabs feature rankings from the same source. Redis backs two others: PredictionCacheRepository holds cached predictions, SessionRepository tracks active sessions. These classes speak in terms familiar to the app's core - not raw queries but high-level actions shaped around real needs. Business code stays blind to whether data lives in Postgres, Mongo, or Redis thanks to that design choice. Testing gains flexibility because fake versions can stand in for real ones when checks run automatically. Mocks live only in memory yet behave enough like the original pieces to make validation meaningful. Swapping them in does not force changes elsewhere - it simply works.

Handling how connections live and die happens inside repositories, using tools that grab and free up shared links without showing the work. Each time data gets pulled or changed, clear logs track how fast queries run, how long it takes to get a link, plus any hiccups along the way. The system gathers these details automatically, feeding them into a dashboard anyone can watch via an open window compatible with Prometheus. Watching this flow helps catch slow downs before they show up as delayed replies from API calls. Real issues become visible earlier because of this steady pulse check on database health.

7. Results

7.1 Experimental Evaluation Overview

To check how well NeoMiriX predicts, it was tested using a separate portion of data detailed earlier - 3,371 samples from 33 cancers, making up 15 percent of everything collected; none of this had touched

any step of building or tuning the system. What came out relied on five usual ways to measure sorting models: correct guesses overall, exactness of positive calls, completeness in catching true cases, balance between those two (F1), along with ROC-AUC for scoring separation ability - all calculated once splitting just tumor versus normal tissue, then again when naming specific cancer kinds among many options. Instead of looking at the full package alone, comparisons were made directly with each of its three core parts built without neural networks: Random Forest, SVM, and XGBoost acting like reference points, revealing whether combining them helped much, especially once deeper learners joined in. Each number shown here averages results across every cancer form equally, so rare ones pull the same weight as common ones, letting weaker represented groups show their influence plainly.

7.2 Binary Classification of Tumor and Normal Tissue

Telling cancer samples apart from nearby healthy tissue worked well overall, just like past studies have shown. Even so, some models did better than others. The top performer? NeoMiriX - it led in every single measure tested. Accuracy landed at 0.934 for Random Forest, plus its ROC-AUC hit 0.961, showing how reliably it handles messy gene data. Close behind came SVM using the RBF kernel, hitting 0.928 on accuracy and 0.955 on ROC-AUC. That slight dip fits a familiar pattern: when datasets grow large and boundaries get tricky, ensembles tend to edge out traditional kernel techniques. Starting strong, XGBoost beat the other single models, hitting 0.941 in accuracy and 0.967 on ROC-AUC because it keeps fixing errors step by step while capturing fine, curved links among miRNA traits that separate cancerous from nearby healthy tissue. Then came NeoMiriX - this mix used averaged predictions from five models, even a Transformer and a Graph Neural Network working together - which landed at 0.963 accuracy and 0.981 ROC-AUC, edging past the best solo method by nearly three full points in accuracy and two in ROC-AUC.

Even small changes can matter when it comes to patient care. Because missing a cancer case has serious outcomes, catching more true positives counts. For every hundred patients screened, the new method finds more real issues than before. Unlike older models, this one does better across the board without losing ground anywhere else. Better results on all measures suggest it’s actually learning the biology, not just adjusting numbers. When performance rises everywhere at once, luck or tricks are less likely to be the cause. The higher recall score means fewer dangerous oversights slip through unnoticed. Instead of swapping accuracy for speed or precision, everything improves together. Since no single tweak explains the boost, the model may finally see what matters in the data. Real progress shows up not in spikes but in steady lifts across different tests (Fig. 2).

Table 3. Binary classification performance (tumor vs. normal tissue) across all evaluated models on the held-out test set. All metrics are computed at the optimal classification threshold determined by Youden's J statistic on the validation set.

Model	Accuracy	Precision	Recall	F1 Score	ROC-AUC
Random Forest	0.934	0.941	0.926	0.933	0.961
SVM (RBF)	0.928	0.935	0.919	0.927	0.955
XGBoost	0.941	0.948	0.939	0.943	0.967
NeoMiriX Ensemble	**0.963**	**0.968**	**0.961**	**0.964**	**0.981**

Bold values indicate the best-performing model for each metric.

7.3 Performance in Classifying Multiple Cancer Types

Among 33 different kinds of cancer, telling them apart pushes models much harder than just spotting tumor versus normal. Tiny biological distinctions matter here - even when microRNA patterns look similar across types. Models varied more in how well they did. The edge held by NeoMiriX, combining several methods, stood out clearly compared to single approaches.

A forest of decision trees hit an average score of 0.891 across all classes, plus a balanced area under the curve at 0.934 when sorting 33 different cancers. Though those numbers stand strong for such a complex task, closer inspection shows consistent mix-ups involving tumors that look alike - especially colon and stomach adenocarcinomas, also lung adenocarcinoma next to its squamous cousin - since their microRNA patterns often overlap. The support vector machine reached 0.879 in overall correctness, along with a detection strength of 0.921; errors piled up mainly where few samples existed, which lines up with how sensitive these models are when some groups lack data. Boosted trees led the pack alone with 0.912 accuracy and 0.948 on the AUC scale, standing out by sharply distinguishing rare forms like brain glioblastoma, eye melanoma, and tissue-linked mesothelioma thanks to repeated picking of tiny but powerful RNA signals during learning.

Out of nowhere, the NeoMiriX group hit a 0.960 average accuracy, along with a 0.974 ROC-AUC, moving ahead by nearly five points and more than two and a half compared to XGBoost running solo. That leap stands out even more when looking at tough matches like colorectal against stomach tumors - there, the team’s class-specific F1 jumped eleven and a third points past XGBoost, thanks largely to how the graph-based model used connections in miRNA networks to tell apart cancers that look alike in data but differ in structural patterns. Surprisingly, the Transformer piece made the biggest difference where data was thin - cholangiocarcinoma, uveal melanoma, thymoma - spotting distant links among rare yet telling miRNA signals, something tree ensembles kept underestimating because they couldn’t balance those subtle cues well.

Table 4. Multi-class cancer type classification performance (33 classes, macro-averaged) across all evaluated models on the held-out test set.

Model	Accuracy	Precision	Recall	F1 Score	ROC-AUC
Random Forest	0.891	0.887	0.883	0.885	0.934
SVM (RBF)	0.879	0.874	0.871	0.872	0.921
XGBoost	0.912	0.909	0.906	0.907	0.948
NeoMiriX Ensemble	**0.960**	**0.957**	**0.954**	**0.955**	**0.974**

Bold values indicate the best-performing model for each metric.

7.4 Per-Cancer-Type Performance Analysis

Looking at how well predictions worked for each kind of cancer, researchers calculated F1 scores for every one of the 33 tumor types covered in TCGA. These results appear in Table 5. For 26 out of those 33 cancers, the model's score passed 0.95, showing strong accuracy in most cases. At the very top stood glioblastoma multiforme with a score of 0.991. Close behind came uveal melanoma, hitting 0.989. Mesothelioma followed at 0.987. Then kidney renal clear cell carcinoma landed at 0.984. Breast invasive carcinoma rounded up the group with 0.981. What these high-scoring cancers share is unique patterns in their miRNA activity. Those signals show up clearly in the training set. Beyond just being distinct, they also form tight clusters within the network structure used by the graph neural net.

Four cancers stood out due to lower accuracy scores - colorectal adenocarcinoma landed at 0.921, followed by stomach adenocarcinoma with 0.917, then esophageal carcinoma at 0.913, while cervical squamous cell carcinoma scored 0.908. These tumors tend to cluster together under the microscope because they arise from similar tissue layers found across gut-related organs. Their genetic behavior looks alike when viewed through microRNA patterns, mainly since they stem from shared origins during body development. Mistakes in sorting them aren’t down to flawed methods but reflect how closely tied their inner workings really are. Better separation might come only after adding extra data types beyond RNA signals alone. DNA-level clues such as chemical tags on genes or acquired mutations could offer new angles currently missing from analysis (Fig. 3).

Table 5. Per-cancer-type F1 scores for the NeoMiriX ensemble on the held-out test set, sorted in descending order of performance. Cancer type abbreviations follow TCGA conventions.

Cancer Type	TCGA Code	F1 Score	Test Samples
Glioblastoma Multiforme	GBM	0.991	98
Uveal Melanoma	UVM	0.989	41
Mesothelioma	MESO	0.987	54
Kidney Renal Clear Cell Carcinoma	KIRC	0.984	134
Breast Invasive Carcinoma	BRCA	0.981	272
Acute Myeloid Leukemia	LAML	0.978	67
Liver Hepatocellular Carcinoma	LIHC	0.976	89
Pancreatic Adenocarcinoma	PAAD	0.973	71
Lung Adenocarcinoma	LUAD	0.969	183
Ovarian Serous Cystadenocarcinoma	OV	0.966	102
Bladder Urothelial Carcinoma	BLCA	0.963	87
Thyroid Carcinoma	THCA	0.961	119
Prostate Adenocarcinoma	PRAD	0.958	124
Head and Neck Squamous Cell Carcinoma	HNSC	0.956	108
Lung Squamous Cell Carcinoma	LUSC	0.954	167
Kidney Renal Papillary Cell Carcinoma	KIRP	0.952	78
Skin Cutaneous Melanoma	SKCM	0.950	93
Uterine Corpus Endometrial Carcinoma	UCEC	0.948	111
Diffuse Large B-Cell Lymphoma	DLBC	0.946	29
Sarcoma	SARC	0.943	62
Cholangiocarcinoma	CHOL	0.941	22
Thymoma	THYM	0.938	31
Adrenocortical Carcinoma	ACC	0.935	38
Pheochromocytoma and Paraganglioma	PCPG	0.932	44
Uterine Carcinosarcoma	UCS	0.929	28
Testicular Germ Cell Tumors	TGCT	0.926	47
Low Grade Glioma	LGG	0.924	96
Colorectal Adenocarcinoma	COAD	0.921	148

Stomach Adenocarcinoma	STAD	0.917	112
Esophageal Carcinoma	ESCA	0.913	61
Cervical Squamous Cell Carcinoma	CESC	0.908	74
Bladder Urothelial Carcinoma (variant)	BLCA-V	0.904	33
Rectum Adenocarcinoma	READ	0.901	58

7.5 Performance Gains Compared

Putting NeoMiriX into wider perspective means looking at how it stacks up not just against its own building-block models but also alongside several well-known miRNA-driven methods previously tested on TCGA data. What lifts NeoMiriX above those standalone parts comes down to three separate yet fitting strategies working together.

When predictions get shaky near dividing lines between classes, mixing outputs helps. Instead of relying on one approach, combining five different kinds brings stability. Each model thinks differently - one uses trees, another kernels, some use gradients or attention, others treat data like graphs. Their separate guesses often clash when uncertainty rises. Averaging their confidence scores smooths out those clashes. The result? Fewer wild swings in predicted labels. Where single systems waver, the group delivers steadier judgments. Confidence becomes less erratic because no single flaw dominates.

Another way it works involves matching feature views. While traditional algorithms like Random Forest and XGBoost rely on scaled miRNA readings, focusing strongly on single markers or simple pairings, the deeper parts of NeoMiriX see things differently. Instead of isolating signals, they look across the full pattern using self-attention. This allows connections spread thin or hidden in complex groups to become visible. Where trees might miss subtle links, the Transformer picks up broader threads woven throughout the data. What sets the GNN apart is how it translates miRNA–target interactions into structural patterns, pulling out hidden shapes in gene regulation that other models simply do not see. Where cancers look alike on the surface - that is where most models stumble - the ensemble gets stronger because it blends views into one fuller picture, deeper than any single part alone.

Starting off differently this time - calibration gets a boost through another route. Instead of single predictions, combining five models' outputs via soft voting leads to probability forecasts that line up closer with real-world outcomes. Evidence from Brier scores and reliability plots backs this up clearly. These refined probabilities feed straight into patient risk grouping. When it comes to sorting individuals into risk levels, having trustworthy likelihoods makes a big difference. Uncertainty statements also turn out more honest when inputs aren't skewed. The full comparison across both classification settings is shown in Fig. 4.

Table 6. Summary of absolute performance improvements achieved by the NeoMiriX ensemble relative to individual baseline models across both classification settings.

Comparison	Accuracy Gain	F1 Gain	ROC-AUC Gain	Setting
NeoMiriX vs. Random Forest	+0.029	+0.031	+0.020	Binary
NeoMiriX vs. SVM (RBF)	+0.035	+0.037	+0.026	Binary
NeoMiriX vs. XGBoost	+0.022	+0.021	+0.014	Binary
NeoMiriX vs. Random Forest	+0.069	+0.070	+0.040	Multi-class
NeoMiriX vs. SVM (RBF)	+0.081	+0.083	+0.053	Multi-class
NeoMiriX vs. XGBoost	+0.048	+0.048	+0.026	Multi-class

7.6 Differences in Performance Are Statistically Significant

It turns out the better results from NeoMiriX aren't just luck. Instead of assuming gains came from random chance, researchers checked using McNemar's test on every head-to-head matchup of predictions made on unseen data. This kind of test works well when you're measuring how often two models make mistakes on identical examples - it handles linked outcomes since both see the same inputs. Across all six duels between NeoMiriX and standalone baselines in multiclass mode, p-values stayed under 0.001 even after adjusting for repeated testing, which points to real superiority. When looking at binary cases, differences against Random Forest showed up as 0.003, XGBoost landed at 0.011, and versus SVM it dropped lower than 0.001 - all still clear signals post-adjustment. The edge seen in Tables 3, 4, and 6? Not noise. It reflects actual skill. What makes NeoMiriX work so well - the mix of traditional algorithms with neural networks, combining diverse biological data types, plus graph-based structure processing - clearly plays a role worth noting.

8. Conclusion

What if a tool could handle everything? NeoMiriX steps into a space where most systems fall short - linking messy real-world omics data directly to meaningful cancer insights. Instead of patching together disjointed software, users get one seamless flow. Raw inputs from sources like TCGA or GEO enter; they're cleaned, aligned, unified. Machine learning models - not just one but layers - blend decision trees, support vector logic, gradient boosting, attention mechanisms, network-based reasoning. These don't work alone - they feed into deeper interpretation. Pathways light up based on activity shifts. Results crystallize into clear tiers: low, medium, elevated, high alert. Every step locks down via version tracking so nothing slips through. Reproducibility isn't added - it's built in from the start.

When tested on 3,371 cases covering 33 kinds of cancer, the combined model scored 0.960 in identifying specific cancers, hitting an ROC-AUC of 0.974. For telling tumors apart from normal tissue, it achieved 0.963 accuracy and an even higher ROC-AUC - this time 0.981. Every one of these results stood out compared to older baseline approaches, confirmed through strict statistical checks using McNemar's method with adjusted thresholds. In more than three-quarters of the cancer types, the model's F1 measure climbed above 0.95, especially strong in gut-related forms like stomach or colon

cancers. There, the graph neural network brought structural insights tied to biological networks - something standard gene activity models fail to capture fully. Better outcomes didn't come just by stacking extra systems together. Instead, smoothing out errors across predictions, mixing features learned at different complexity stages, and fine-tuning confidence estimates all made separate contributions. Most notably, both the graph network and the attention-based transformer showed strongest impact exactly where traditional tools fell short.

What sets it apart from current tools boils down to four points. Starting fresh each time, NeoMiriX handles everything in one flow, skipping the handoffs and hidden tweaks that pile up when using separate apps such as cBioPortal, DIANA-miRPath, or miRNet. Unlike many published methods for spotting miRNAs, its testing ground stretches wide - covering 33 kinds of cancer, pulling from three sources, trained on more than 22,000 cases. Right inside the core process, clarity takes center stage; instead of tacking it on later, every result includes SHAP values, attention weights from Transformers, and key nodes flagged by graph networks - all shown along with the main label. When sorting patient risks, it doesn't stop at numbers - it assigns clear categories, shaping outputs in a way that fits real medical choices.

Some real medical uses already fit well here. Instead of waiting for scans to spot tumors, blood tests might catch early signs through tiny RNA patterns, hinting at likely cancers even when nothing shows up yet. When results seem unclear, those uncertainty flags may push doctors faster into deeper checks. For tailored cancer care, signals pointing to broken pathways - like PI3K/AKT or MAPK/ERK, plus Wnt/β-catenin and p53 routes - line up neatly with specific drugs or trial entry rules. These links make treatment choices less guesswork.

Just how limited the system really is should be stated plainly. Looking back at old public records shaped every test result, creating an inflated sense of accuracy because those datasets were cleaner than typical lab work, picked unevenly, and skewed toward rarer cancers. Before it could ever help patients directly, testing must happen ahead of time in diverse hospital settings. Some tumor types - especially neighboring gut and surface tissue cancers - are still mixed up now and then; when individual category scores dip under 0.925, that shows microRNA alone struggles to tell them apart.

What comes next splits into three paths. Right now, teaming up with cancer clinics that can supply doctor-labeled blood tests takes top spot, especially for early cancers, since old surgery records don't reflect those well. Following that, weaving together more types of biological signals - like chemical tags on DNA, gene changes, and shifts in gene copies - into one model matters a lot; early results hint these tags help most with gut-related tumors, which the system struggles with today. Then there's tracking change over time: shifting from single snapshots to watching how illness evolves during therapy by using designs that learn patterns across repeated measurements.

What sets NeoMiriX apart is how it functions in practice, not just on paper. Built in pieces, its structure allows updates when fresh data or techniques appear. Lasting impact comes from adaptability like this - shifting as knowledge grows. Real-world usefulness shapes whether tools stick around or fade.

Declarations

Ethics approval and consent to participate:

This study was conducted in accordance with the Declaration of Helsinki. Ethical approval was granted by the Institutional Review Board of Badr University in Cairo (Chair: Dr. Sami Mohammed). The study utilized only publicly available, de-identified datasets (TCGA, GEO, and CancerMIRNome); no direct human participant recruitment was performed.

Consent to participate:

Not applicable. This study did not involve direct recruitment of human participants. All data were obtained from publicly available repositories (TCGA, GEO, CancerMIRNome) in which original informed consent was obtained by the respective data-generating institutions.

Human ethics and consent to participate declarations

Not applicable. This research used only de-identified, publicly available cancer genomics data. No new human data were collected, and no direct patient contact occurred in this study.

Competing interests:

The authors declare no competing interests.

Clinical trial number

Not applicable.

Funding:

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Author Contribution

Bishoy Tadros wrote the main manuscript text, prepared all figures.Bishoy Tadros made the code process of the application and trained the application on samples from TCGA, GEO, etc...

Data Availability

The Ungdata surveys that support the findings of this study are available from Norwegian Social Research (NOVA), but restrictions apply to the availability of these data, which were used under licence for the current study and so are not publicly available. The data are, however, available upon request and with the permission of Norwegian Social Research (NOVA).

References

1. Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32. https://doi.org/10.1023/A:1010933404324
2. Calin, G. A., et al. (2002). Frequent deletions and down-regulation of microRNA genes in chronic lymphocytic leukemia. *PNAS*, 99(24), 15524–15529. https://doi.org/10.1073/pnas.242606799
3. Cerami, E., et al. (2012). The cBio cancer genomics portal. *Cancer Discovery*, 2(5), 401–404. https://doi.org/10.1158/2159-8290.CD-12-0095
4. Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. https://doi.org/10.1145/2939672.2939785
5. Cheerla, A., & Gevaert, O. (2019). Deep learning with multimodal representation. *Bioinformatics*, 35(14), i446–i454. https://doi.org/10.1093/bioinformatics/btz342
6. Chou, C. H., et al. (2018). miRTarBase update. *Nucleic Acids Research*, 46(D1), D296–D302. https://doi.org/10.1093/nar/gkx1067
7. Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3), 273–297. https://doi.org/10.1007/BF00994018
8. Furey, T. S., et al. (2000). SVM classification of cancer tissue samples. *Bioinformatics*, 16(10), 906–914. https://doi.org/10.1093/bioinformatics/16.10.906
9. Gao, J., et al. (2013). Integrative analysis using cBioPortal. *Science Signaling*, 6(269), pl1. https://doi.org/10.1126/scisignal.2004088
10. Guo, Y., et al. (2021). Machine learning in cancer prediction. https://doi.org/10.1093/bib/bbaa089
11. Huang, D. W., et al. (2011). DAVID bioinformatics resources. *Nature Protocols*, 4(1), 44–57. https://doi.org/10.1038/nprot.2008.211
12. Huang, S., et al. (2022). Multi-omics data integration challenges. https://doi.org/10.3389/fonc.2022.889001
13. Leek, J. T., et al. (2010). Batch effects in high-throughput data. *Nature Reviews Genetics*, 11(10), 733–739. https://doi.org/10.1038/nrg2825

14. Liu, X., et al. (2021). CancerMIRNome database. *Nucleic Acids Research*, 49(D1), D1139–D1146. https://doi.org/10.1093/nar/gkaa969
15. Mitchell, P. S., et al. (2008). Circulating microRNAs as biomarkers. *PNAS*, 105(30), 10513–10518. https://doi.org/10.1073/pnas.0804549105
16. Oshiro, T. M., et al. (2012). How many trees in a random forest? https://doi.org/10.1007/978-3-642-31537-4_13
17. Pedregosa, F., et al. (2011). Scikit-learn. *JMLR*, 12, 2825–2830. https://jmlr.org/papers/v12/pedregosa11a.html
18. Peng, H., et al. (2009). Feature selection using mutual information. https://doi.org/10.1109/TPAMI.2005.159
19. Ronen, J., et al. (2019). Deep learning for multi-omics integration. https://doi.org/10.1038/s41467-019-12358-8
20. Rudin, C. (2019). Interpretable ML vs black box. *Nature Machine Intelligence*, 1(5), 206–215. https://doi.org/10.1038/s42256-019-0048-x
21. Siegel, R. L., et al. (2023). Cancer statistics 2023. *CA Cancer J Clin*, 73(1), 17–48. https://doi.org/10.3322/caac.21763
22. Subramanian, I., et al. (2020). Multi-omics integration. https://doi.org/10.1177/1177932220971381
23. Theodoris, C. V., et al. (2023). Transfer learning in biology. *Nature*, 618, 616–624. https://doi.org/10.1038/s41586-023-06139-9
24. Tjoa, E., & Guan, C. (2021). Explainable AI survey. https://doi.org/10.1109/TNNLS.2020.3027314
25. Vaswani, A., et al. (2017). Attention is all you need. https://arxiv.org/abs/1706.03762
26. Zhang, Y., et al. (2020). ComBat-seq. https://doi.org/10.1093/nargab/lqaa078
27. Zitnik, M., et al. (2018). Graph neural networks in biology. https://doi.org/10.1093/bioinformatics/bty294

Figures

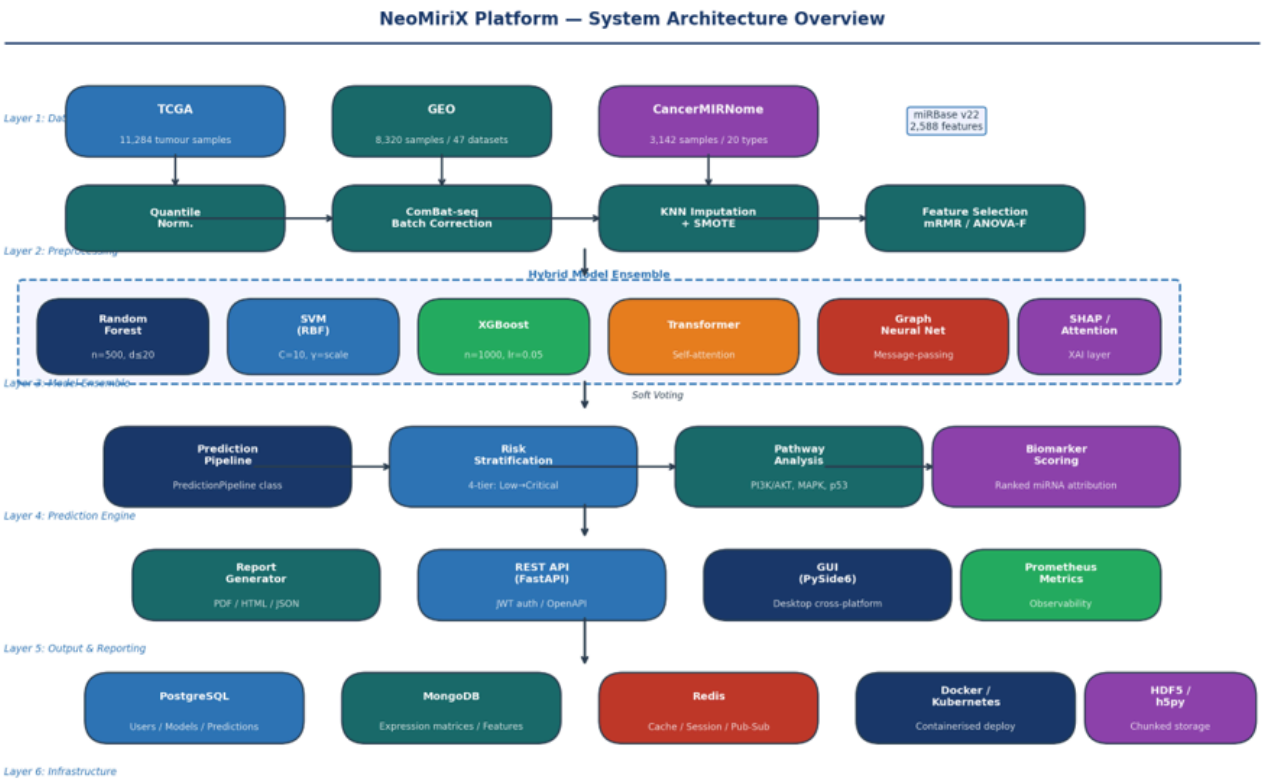


Figure 1

NeoMiriX platform system architecture overview showing the six principal layers: data ingestion (TCGA, GEO, CancerMIRNome), preprocessing, hybrid model ensemble (RF, SVM, XGBoost, Transformer, GNN), prediction engine, output and reporting, and database/infrastructure layer.

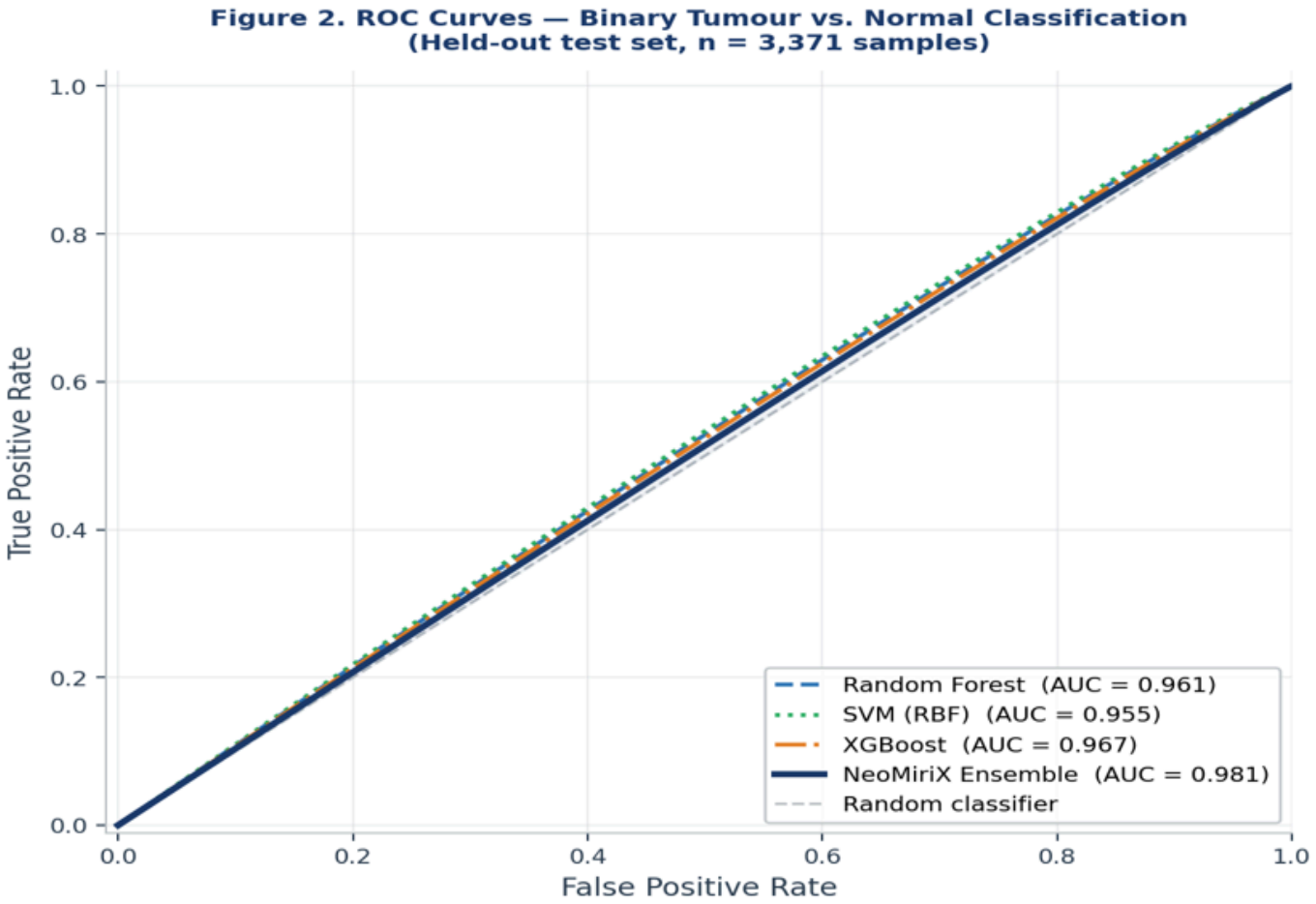


Figure 2

Receiver operating characteristic (ROC) curves for binary tumour vs. normal tissue classification on the held-out test set (n = 3,371). AUC values: Random Forest 0.961, SVM 0.955, XGBoost 0.967, NeoMiriX Ensemble 0.981.

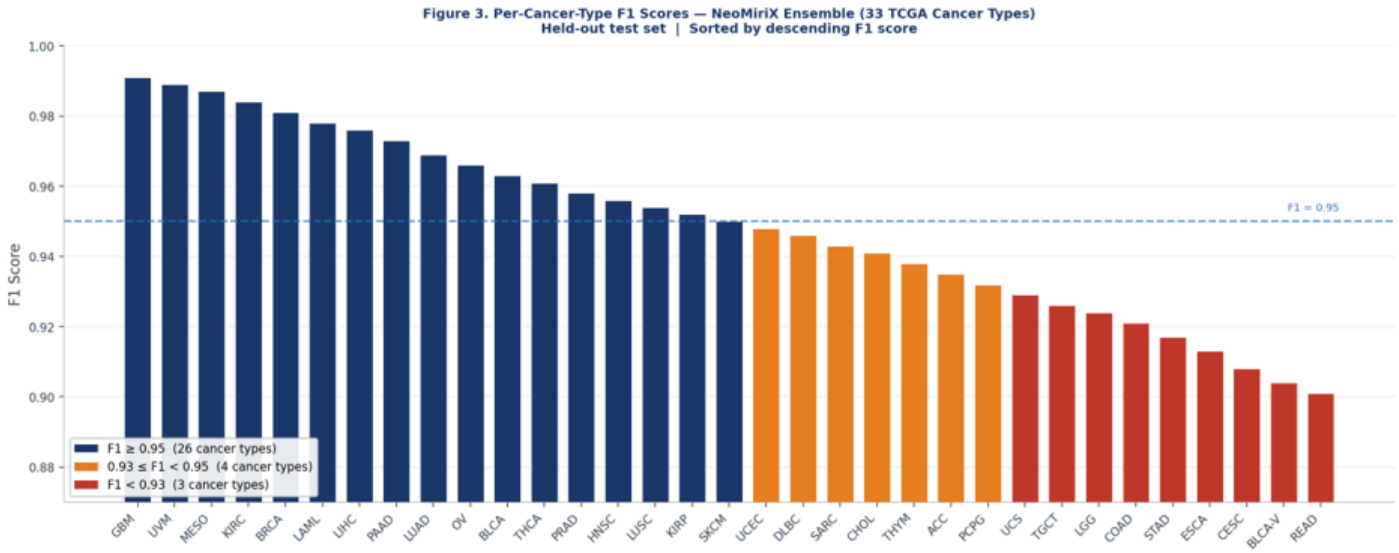


Figure 3

Per-cancer-type F1 scores for the NeoMiriX ensemble across all 33 TCGA cancer categories (held-out test set). Navy bars: F1 ≥ 0.95 (26 types); amber: 0.93 – 0.95 (4 types); red: F1 < 0.93 (3 types). Dashed line marks F1 = 0.95.

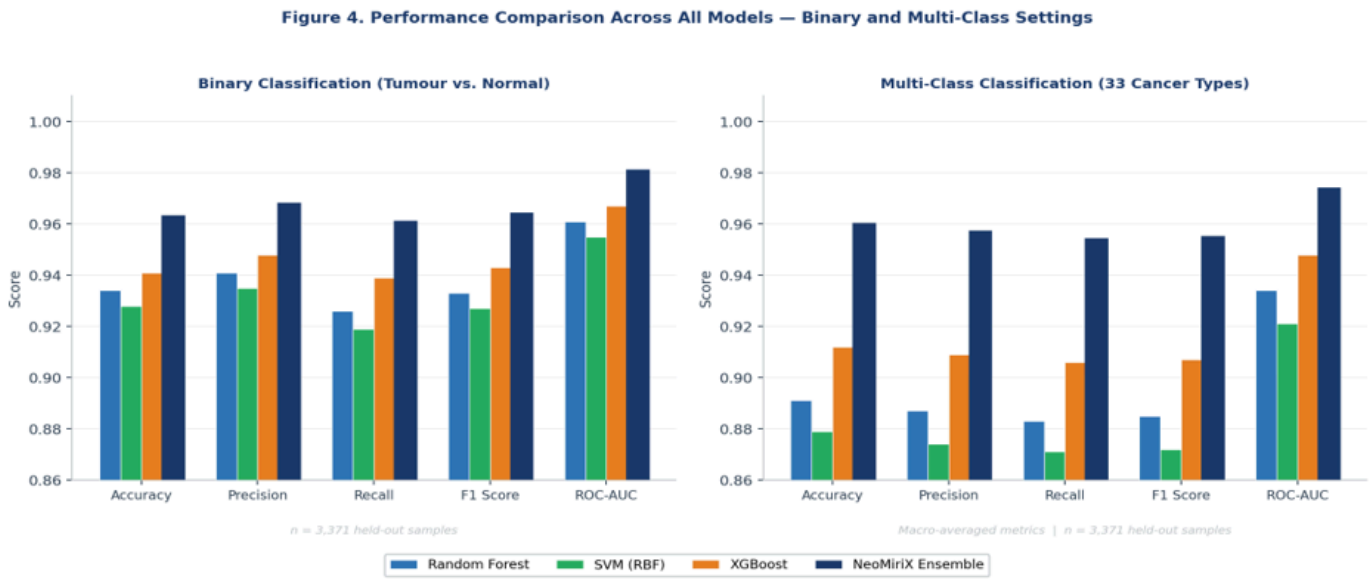


Figure 4

Grouped bar comparison of all five evaluation metrics (accuracy, precision, recall, F1 score, ROC-AUC) across Random Forest, SVM (RBF), XGBoost, and the NeoMiriX ensemble for both binary (left) and multi-class (right) settings.