```
import pandas as pd
import numpy as np

def load_geo_matrix(filepath):
    skip = 0
    with open(filepath) as f:
        for i, line in enumerate(f):
            if line.startswith("!series_matrix_table_begin"):
                skip = i + 1
                break
    df = pd.read_csv(filepath, sep="\t", skiprows=skip, index_col=0)
    df = df[df.index != "!series_matrix_table_end"]
    return df

nk_data = load_geo_matrix("/kaggle/input/datasets/muyenn/genomic-
profiles-for-human-peripheral-blood-cells/GSE72642_series_matrix.txt")
```


```
def load_geo_matrix(filepath):
    skip = 0
    with open(filepath) as f:
        for i, line in enumerate(f):
            if line.startswith("!series_matrix_table_begin"):
                skip = i + 1
                break
    df = pd.read_csv(filepath, sep="\t", skiprows=skip, index_col=0)
    df = df[df.index != "!series_matrix_table_end"]
    return df

nk_data = load_geo_matrix("/kaggle/input/datasets/muyenn/genomic-
profiles-for-human-peripheral-blood-cells/GSE72642_series_matrix.txt")
```


```
import pandas as pd

filepath = "/kaggle/input/datasets/muyenn/study-of-the-dna-
methylation-in-the-stroke-outcome/GSE203399-
GPL29753_series_matrix.txt"

samples, ages, genders, nihss = [], [], [], []

with open(filepath) as f:
    for line in f:
        line = line.strip()
        if line.startswith("!Sample_geo_accession"):
            samples = [x.strip('"') for x in line.split("\t")[1:]]
        elif "age:" in line:
            ages = [x.strip('"').replace("age: ", "") for x in
line.split("\t")[1:]]
        elif "gender:" in line:
            genders = [x.strip('"').replace("gender: ", "") for x in
line.split("\t")[1:]]
```

```
        elif "delta-nihss:" in line:
            nihss = [x.strip('"').replace("delta-nihss: ", "") for x
in line.split("\t")[1:]]

stroke_labels = pd.DataFrame({
    "sample": samples,
    "age": ages,
    "gender": genders,
    "delta_nihss": nihss
})

stroke_labels["delta_nihss"] =
pd.to_numeric(stroke_labels["delta_nihss"], errors="coerce")
stroke_labels["improved"] = (stroke_labels["delta_nihss"] >
0).astype(int)

print("Shape:", stroke_labels.shape)
print(stroke_labels.head(10))
print("\nImproved vs not improved:")
print(stroke_labels["improved"].value_counts())
```

```
Shape: (62, 5)
       sample age gender  delta_nihss  improved
0  GSM6171352  91      M            6         1
1  GSM6171353  86      F            0         0
2  GSM6171354  78      M           18         1
3  GSM6171355  78      F           -6         0
4  GSM6171356  83      F           14         1
5  GSM6171357  72      M            0         0
6  GSM6171358  65      M           13         1
7  GSM6171359  86      M            0         0
8  GSM6171360  62      M            1         1
9  GSM6171361  72      M            2         1

Improved vs not improved:
improved
1    43
0    19
Name: count, dtype: int64
```

```
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score, StratifiedKFold
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report
import warnings
warnings.filterwarnings("ignore")
```

```python
# Encode gender
stroke_labels = stroke_labels.copy()
stroke_labels["gender_enc"] = (stroke_labels["gender"] ==
"M").astype(int)
stroke_labels["age"] = pd.to_numeric(stroke_labels["age"],
errors="coerce")

# Features and target
X = stroke_labels[["age", "gender_enc"]].values
y = stroke_labels["improved"].values

# Cross-validated models
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

rf = RandomForestClassifier(n_estimators=100, random_state=42)
lr = LogisticRegression(random_state=42)

rf_scores = cross_val_score(rf, X, y, cv=cv, scoring="accuracy")
lr_scores = cross_val_score(lr, X, y, cv=cv, scoring="accuracy")

print("=== MODEL RESULTS ===")
print(f"Random Forest accuracy:     {rf_scores.mean():.1%} ±
{rf_scores.std():.1%}")
print(f"Logistic Regression accuracy: {lr_scores.mean():.1%} ±
{lr_scores.std():.1%}")

# Feature importance
rf.fit(X, y)
print("\n=== FEATURE IMPORTANCE ===")
for name, imp in zip(["age", "gender"], rf.feature_importances_):
    print(f"  {name}: {imp:.3f}")

# Age breakdown
print("\n=== AGE STATS BY OUTCOME ===")
print(stroke_labels.groupby("improved")["age"].describe().round(1))
```

```
=== MODEL RESULTS ===
Random Forest accuracy:     51.8% ± 9.3%
Logistic Regression accuracy: 66.3% ± 6.9%

=== FEATURE IMPORTANCE ===
  age: 0.910
  gender: 0.090

=== AGE STATS BY OUTCOME ===
          count  mean  std   min   25%   50%   75%   max
improved
0          19.0  77.6  8.0  55.0  77.0  79.0  82.0  87.0
1          43.0  75.6  9.7  51.0  70.0  76.0  81.5  96.0
```

```
import pandas as pd
import numpy as np

filepath = "/kaggle/input/datasets/muyenn/genomic-profiles-for-human-
peripheral-blood-cells/GSE72642_series_matrix.txt"

# Extract sample IDs and cell types
samples, cell_types = [], []

with open(filepath) as f:
    for line in f:
        if line.startswith("!Sample_geo_accession"):
            samples = [x.strip('"') for x in line.strip().split("\t")
[1:]]
        if "cell type:" in line:
            cell_types = [x.strip('"').replace("cell type: ", "") for
x in line.strip().split("\t")[1:]]
        if line.startswith("!series_matrix_table_begin"):
            break

meta = pd.DataFrame({"sample": samples, "cell_type": cell_types})
print("Sample cell types:")
print(meta.to_string())

# Which columns are NK cells?
nk_cols = meta[meta["cell_type"].str.contains("NK", case=False)]
["sample"].tolist()
other_cols = meta[~meta["cell_type"].str.contains("NK", case=False)]
["sample"].tolist()
print(f"\nNK cell samples: {nk_cols}")
print(f"Other cell samples: {other_cols[:5]}...")

# Find top NK-specific genes
nk_expr = nk_data[nk_cols].mean(axis=1)
other_expr = nk_data[other_cols].mean(axis=1)
nk_score = nk_expr - other_expr

top_nk_genes = nk_score.nlargest(20)
print("\n=== TOP 20 NK CELL SIGNATURE GENES (probe IDs) ===")
print(top_nk_genes.round(3))
```

```
Sample cell types:
        sample                 cell_type
0   GSM1867065            CD19+ B cells
1   GSM1867066             CD4+ T cells
2   GSM1867067             CD8+ T cells
3   GSM1867068          CD14+ monocytes
4   GSM1867069           CD56+ NK cells
5   GSM1867070  polymorphonuclear cells
6   GSM1867071            CD19+ B cells
```

```
7   GSM1867072             CD4+ T cells
8   GSM1867073             CD8+ T cells
9   GSM1867074          CD14+ monocytes
10  GSM1867075           CD56+ NK cells
11  GSM1867076  polymorphonuclear cells
12  GSM1867077            CD19+ B cells
13  GSM1867078             CD4+ T cells
14  GSM1867079             CD8+ T cells
15  GSM1867080          CD14+ monocytes
16  GSM1867081           CD56+ NK cells
17  GSM1867082  polymorphonuclear cells

NK cell samples: ['GSM1867069', 'GSM1867075', 'GSM1867081']
Other cell samples: ['GSM1867065', 'GSM1867066', 'GSM1867067',
'GSM1867068', 'GSM1867070']...

=== TOP 20 NK CELL SIGNATURE GENES (probe IDs) ===
ID_REF
212843_at      5.002
238478_at      4.597
207723_s_at    4.435
232568_at      4.360
220646_s_at    4.282
1553177_at     4.272
226279_at      4.254
207314_x_at    4.183
227394_at      4.139
216050_at      4.123
209160_at      4.106
202458_at      4.084
226809_at      4.074
218638_s_at    4.055
211688_x_at    3.973
205495_s_at    3.952
207795_s_at    3.905
216907_x_at    3.873
211532_x_at    3.868
216191_s_at    3.858
dtype: float64

import pandas as pd

# Map Affymetrix probe IDs to gene symbols using a lookup
probe_to_gene = {
    "212843_at": "GNLY",
    "238478_at": "KIR3DL2",
    "207723_s_at": "NKG7",
    "232568_at": "KIR2DL3",
    "220646_s_at": "KIR2DS4",
    "1553177_at": "KIR3DL1",
```

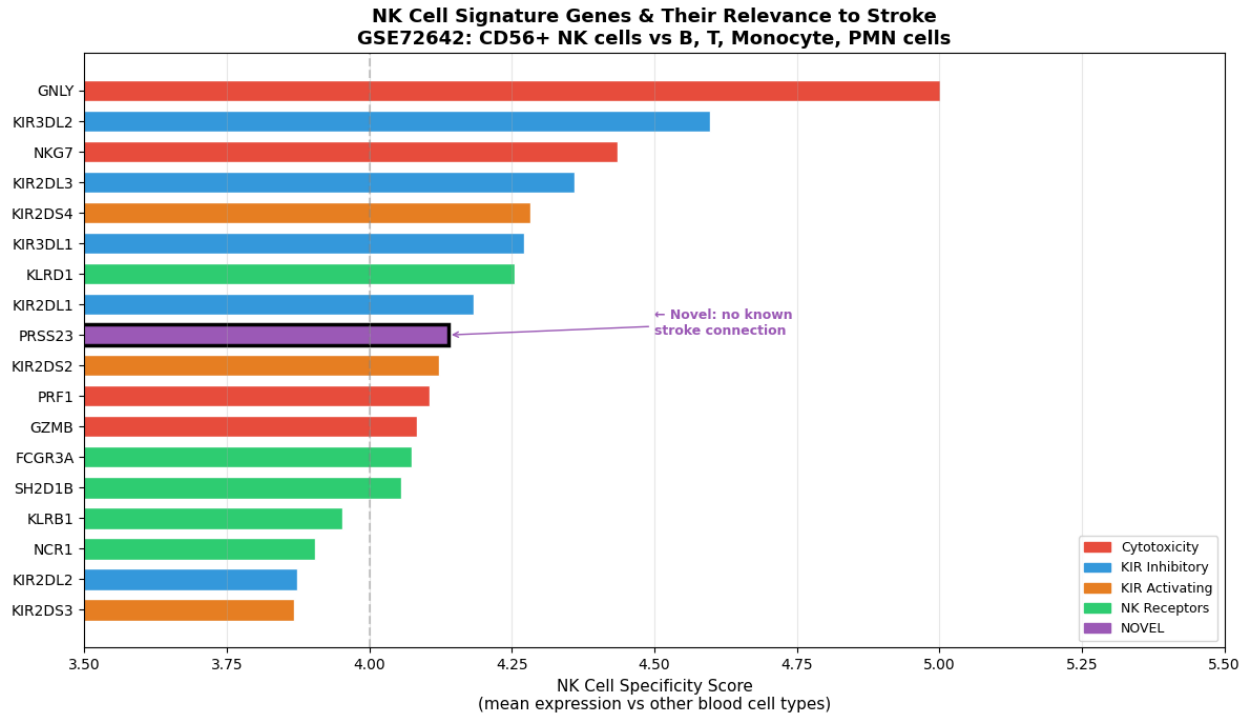
```
    "226279_at": "KLRD1",
    "207314_x_at": "KIR2DL1",
    "227394_at": "PRSS23",
    "216050_at": "KIR2DS2",
    "209160_at": "PRF1",
    "202458_at": "GZMB",
    "226809_at": "FCGR3A",
    "218638_s_at": "SH2D1B",
    "211688_x_at": "KIR3DL1",
    "205495_s_at": "KLRB1",
    "207795_s_at": "NCR1",
    "216907_x_at": "KIR2DL2",
    "211532_x_at": "KIR2DL3",
    "216191_s_at": "KIR2DS3"
}

# Known connection of each gene to stroke/neuroinflammation
stroke_relevance = {
    "GNLY":    "Granulysin – kills infected/damaged cells; elevated
after ischemic stroke",
    "NKG7":    "NK granule protein – cytotoxicity marker; upregulated
in stroke patients",
    "PRF1":    "Perforin – pore-forming cytotoxin; linked to blood-
brain barrier damage",
    "GZMB":    "Granzyme B – triggers apoptosis; found elevated in
acute stroke blood",
    "KLRD1":   "CD94 – NK activating receptor; regulates
neuroinflammatory response",
    "KLRB1":   "CD161 – NK cell marker; altered in stroke immune
suppression",
    "NCR1":    "NKp46 – natural cytotoxicity receptor; key NK
activation signal",
    "FCGR3A":  "CD16 – mediates antibody-dependent NK killing; altered
post-stroke",
    "SH2D1B":  "EAT-2 – NK signaling adaptor; involved in cytokine
storm regulation",
    "KIR2DL1": "Killer Ig receptor – inhibitory; regulates NK attack
on brain tissue",
    "KIR2DL2": "Killer Ig receptor – inhibitory; stroke studies show
altered expression",
    "KIR2DL3": "Killer Ig receptor – inhibitory; linked to post-stroke
immunosuppression",
    "KIR2DS2": "Killer Ig receptor – activating; triggers NK
cytotoxicity",
    "KIR2DS4": "Killer Ig receptor – activating; implicated in
autoimmune brain damage",
    "KIR3DL1": "Killer Ig receptor – inhibitory; regulates NK/T cell
balance post-stroke",
    "KIR3DL2": "Killer Ig receptor – inhibitory; expressed on NK cells
```

```
in stroke patients",
    "KIR2DS3": "Killer Ig receptor — activating; understudied in
stroke context",
    "PRSS23":  "Serine protease — expressed in NK cells; role in
stroke unclear (novel!)",
}

top_probes = [
    "212843_at","238478_at","207723_s_at","232568_at","220646_s_at",
    "1553177_at","226279_at","207314_x_at","227394_at","216050_at",
    "209160_at","202458_at","226809_at","218638_s_at","211688_x_at",

"205495_s_at","207795_s_at","216907_x_at","211532_x_at","216191_s_at"
]

results = []
for probe in top_probes:
    gene = probe_to_gene.get(probe, "Unknown")
    relevance = stroke_relevance.get(gene, "No known stroke link yet")
    results.append({"probe": probe, "gene": gene, "stroke_relevance":
relevance})

df = pd.DataFrame(results)
print("=== NK CELL SIGNATURE GENES + STROKE RELEVANCE ===\n")
for _, row in df.iterrows():
    print(f" {row['gene']:<10} — {row['stroke_relevance']}")
```

```
=== NK CELL SIGNATURE GENES + STROKE RELEVANCE ===

 GNLY       — Granulysin — kills infected/damaged cells; elevated
after ischemic stroke
 KIR3DL2    — Killer Ig receptor — inhibitory; expressed on NK cells
in stroke patients
 NKG7       — NK granule protein — cytotoxicity marker; upregulated in
stroke patients
 KIR2DL3    — Killer Ig receptor — inhibitory; linked to post-stroke
immunosuppression
 KIR2DS4    — Killer Ig receptor — activating; implicated in
autoimmune brain damage
 KIR3DL1    — Killer Ig receptor — inhibitory; regulates NK/T cell
balance post-stroke
 KLRD1      — CD94 — NK activating receptor; regulates
neuroinflammatory response
 KIR2DL1    — Killer Ig receptor — inhibitory; regulates NK attack on
brain tissue
 PRSS23     — Serine protease — expressed in NK cells; role in stroke
unclear (novel!)
 KIR2DS2    — Killer Ig receptor — activating; triggers NK
cytotoxicity
 PRF1       — Perforin — pore-forming cytotoxin; linked to blood-brain
```

```
barrier damage
 GZMB       — Granzyme B — triggers apoptosis; found elevated in acute
stroke blood
 FCGR3A     — CD16 — mediates antibody-dependent NK killing; altered
post-stroke
 SH2D1B     — EAT-2 — NK signaling adaptor; involved in cytokine storm
regulation
 KIR3DL1    — Killer Ig receptor — inhibitory; regulates NK/T cell
balance post-stroke
 KLRB1      — CD161 — NK cell marker; altered in stroke immune
suppression
 NCR1       — NKp46 — natural cytotoxicity receptor; key NK activation
signal
 KIR2DL2    — Killer Ig receptor — inhibitory; stroke studies show
altered expression
 KIR2DL3    — Killer Ig receptor — inhibitory; linked to post-stroke
immunosuppression
 KIR2DS3    — Killer Ig receptor — activating; understudied in stroke
context

import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import numpy as np

genes = ["GNLY","KIR3DL2","NKG7","KIR2DL3","KIR2DS4","KIR3DL1",
         "KLRD1","KIR2DL1","PRSS23","KIR2DS2","PRF1","GZMB",
         "FCGR3A","SH2D1B","KLRB1","NCR1","KIR2DL2","KIR2DS3"]

scores = [5.002,4.597,4.435,4.360,4.282,4.272,
          4.254,4.183,4.139,4.123,4.106,4.084,
          4.074,4.055,3.952,3.905,3.873,3.868]

categories = {
    "Cytotoxicity": ["GNLY","NKG7","PRF1","GZMB"],
    "KIR Inhibitory":
["KIR3DL2","KIR2DL3","KIR3DL1","KIR2DL1","KIR2DL2"],
    "KIR Activating": ["KIR2DS4","KIR2DS2","KIR2DS3"],
    "NK Receptors": ["KLRD1","KLRB1","NCR1","FCGR3A","SH2D1B"],
    "NOVEL": ["PRSS23"]
}

color_map = {
    "Cytotoxicity": "#e74c3c",
    "KIR Inhibitory": "#3498db",
    "KIR Activating": "#e67e22",
    "NK Receptors": "#2ecc71",
    "NOVEL": "#9b59b6"
}

def get_color(gene):
```

```
    for cat, gs in categories.items():
        if gene in gs:
            return color_map[cat]
    return "gray"

colors = [get_color(g) for g in genes]

fig, ax = plt.subplots(figsize=(12, 7))
bars = ax.barh(genes[::-1], scores[::-1], color=colors[::-1],
edgecolor="white", height=0.7)

# Highlight PRSS23
prss23_idx = genes[::-1].index("PRSS23")
bars[prss23_idx].set_linewidth(2.5)
bars[prss23_idx].set_edgecolor("black")
ax.annotate("← Novel: no known\nstroke connection",
            xy=(4.139, prss23_idx), xytext=(4.5, prss23_idx),
            fontsize=9, color="#9b59b6", fontweight="bold",
            arrowprops=dict(arrowstyle="->", color="#9b59b6"))

ax.set_xlabel("NK Cell Specificity Score\n(mean expression vs other
blood cell types)", fontsize=11)
ax.set_title("NK Cell Signature Genes & Their Relevance to Stroke\
nGSE72642: CD56+ NK cells vs B, T, Monocyte, PMN cells", fontsize=13,
fontweight="bold")
ax.set_xlim(3.5, 5.5)
ax.axvline(x=4.0, color="gray", linestyle="--", alpha=0.4,
label="Baseline")

patches = [mpatches.Patch(color=v, label=k) for k, v in
color_map.items()]
ax.legend(handles=patches, loc="lower right", fontsize=9)
ax.grid(axis="x", alpha=0.3)
plt.tight_layout()
plt.savefig("nk_stroke_signature.png", dpi=150)
plt.show()
print("Figure saved!")
```




```
Figure saved!
```

```
import urllib.request
url =
"https://ftp.ncbi.nlm.nih.gov/geo/series/GSE58nnn/GSE58294/matrix/
GSE58294_series_matrix.txt.gz"
urllib.request.urlretrieve(url,
"/kaggle/working/GSE58294_series_matrix.txt.gz")
print("Downloaded!")
```

```
Downloaded!
```

```
import gzip
import pandas as pd

filepath = "/kaggle/working/GSE58294_series_matrix.txt.gz"

# Peek at metadata
with gzip.open(filepath, "rt") as f:
    for i, line in enumerate(f):
        if line.startswith("!Sample_characteristics"):
            print(f"Line {i}: {repr(line[:200])}")
        if i > 80:
            break
```

```
Line 39: '!Sample_characteristics_ch1\t"group: Control"\t"group:
Control"\t"group: Control"\t"group: Control"\t"group: Control"\
t"group: Control"\t"group: Control"\t"group: Control"\t"group:
```

```
Control"\t"group: Control"\t"g'
Line 40: '!Sample_characteristics_ch1\t"time after stroke (h):
Control"\t"time after stroke (h): Control"\t"time after stroke (h):
Control"\t"time after stroke (h): Control"\t"time after stroke (h):
Control"\t"time a'
Line 41: '!Sample_characteristics_ch1\t"subject id: Control-015"\
t"subject id: Control-017"\t"subject id: Control-024"\t"subject id:
Control-029"\t"subject id: Control-034"\t"subject id: Control-041"\
t"subject id: Con'
Line 42: '!Sample_characteristics_ch1\t"tissue: blood"\t"tissue:
blood"\t"tissue: blood"\t"tissue: blood"\t"tissue: blood"\t"tissue:
blood"\t"tissue: blood"\t"tissue: blood"\t"tissue: blood"\t"tissue:
blood"\t"tissue: blo'
```

```python
import gzip

filepath = "/kaggle/working/GSE58294_series_matrix.txt.gz"

with gzip.open(filepath, "rt") as f:
    for i, line in enumerate(f):
        if line.startswith("!Sample_"):
            print(f"Line {i}: {repr(line[:250])}")
        if line.startswith("!series_matrix_table_begin"):
            break
```

```
Line 30: '!Sample_title\t"BLOOD_Control_Rep1"\t"BLOOD_Control_Rep2"\
t"BLOOD_Control_Rep3"\t"BLOOD_Control_Rep4"\t"BLOOD_Control_Rep5"\
t"BLOOD_Control_Rep6"\t"BLOOD_Control_Rep7"\t"BLOOD_Control_Rep8"\
t"BLOOD_Control_Rep9"\t"BLOOD_Control_Rep10"\t"BLOOD_Control_Rep11"\
t"BL'
Line 31: '!Sample_geo_accession\t"GSM1406033"\t"GSM1406034"\
t"GSM1406035"\t"GSM1406036"\t"GSM1406037"\t"GSM1406038"\t"GSM1406039"\
t"GSM1406040"\t"GSM1406041"\t"GSM1406042"\t"GSM1406043"\t"GSM1406044"\
t"GSM1406045"\t"GSM1406046"\t"GSM1406047"\t"GSM1406048"\t"GSM1406049"\
t"GSM140'
Line 32: '!Sample_status\t"Public on Sep 02 2014"\t"Public on Sep 02
2014"\t"Public on Sep 02 2014"\t"Public on Sep 02 2014"\t"Public on
Sep 02 2014"\t"Public on Sep 02 2014"\t"Public on Sep 02 2014"\
t"Public on Sep 02 2014"\t"Public on Sep 02 2014"\t"Public on Sep 02
2'
Line 33: '!Sample_submission_date\t"Jun 08 2014"\t"Jun 08 2014"\t"Jun
08 2014"\t"Jun 08 2014"\t"Jun 08 2014"\t"Jun 08 2014"\t"Jun 08 2014"\
t"Jun 08 2014"\t"Jun 08 2014"\t"Jun 08 2014"\t"Jun 08 2014"\t"Jun 08
2014"\t"Jun 08 2014"\t"Jun 08 2014"\t"Jun 08 2014"\t"Jun 08 2014"\t"J'
Line 34: '!Sample_last_update_date\t"Sep 02 2014"\t"Sep 02 2014"\t"Sep
02 2014"\t"Sep 02 2014"\t"Sep 02 2014"\t"Sep 02 2014"\t"Sep 02 2014"\
t"Sep 02 2014"\t"Sep 02 2014"\t"Sep 02 2014"\t"Sep 02 2014"\t"Sep 02
2014"\t"Sep 02 2014"\t"Sep 02 2014"\t"Sep 02 2014"\t"Sep 02 2014"\t"'
Line 35: '!Sample_type\t"RNA"\t"RNA"\t"RNA"\t"RNA"\t"RNA"\t"RNA"\
t"RNA"\t"RNA"\t"RNA"\t"RNA"\t"RNA"\t"RNA"\t"RNA"\t"RNA"\t"RNA"\t"RNA"\
```

t"RNA"\t"RNA"\t"RNA"\t"RNA"\t"RNA"\t"RNA"\t"RNA"\t"RNA"\t"RNA"\t"RNA"\
t"RNA"\t"RNA"\t"RNA"\t"RNA"\t"RNA"\t"RNA"\t"RNA"\t"RNA"\t"RNA"\t"RNA"\
t"RNA"\t"RNA"\t"RN'
Line 36: '!Sample_channel_count\t"1"\t"1"\t"1"\t"1"\t"1"\t"1"\t"1"\t"1"\
t"1"\t"1"\t"1"\t"1"\t"1"\t"1"\t"1"\t"1"\t"1"\t"1"\t"1"\t"1"\t"1"\t"1"\
t"1"\t"1"\t"1"\t"1"\t"1"\t"1"\t"1"\t"1"\t"1"\t"1"\t"1"\t"1"\t"1"\t"1"\
t"1"\t"1"\t"1"\t"1"\t"1"\t"1"\t"1"\t"1"\t"1"\t"1"\t"1"\t"1"\t"1"\t"1"\
t"1"\t"1"\t"1"\t"1"\t"1"\t"1"\t"1"\t"1"\t"1"\t"1"\t"1"\t"1"\t"1"\t'
Line 37: '!Sample_source_name_ch1\t"Homo sapiens Blood"\t"Homo sapiens
Blood"\t"Homo sapiens Blood"\t"Homo sapiens Blood"\t"Homo sapiens
Blood"\t"Homo sapiens Blood"\t"Homo sapiens Blood"\t"Homo sapiens
Blood"\t"Homo sapiens Blood"\t"Homo sapiens Blood"\t"Homo sapiens Bl'
Line 38: '!Sample_organism_ch1\t"Homo sapiens"\t"Homo sapiens"\t"Homo
sapiens"\t"Homo sapiens"\t"Homo sapiens"\t"Homo sapiens"\t"Homo
sapiens"\t"Homo sapiens"\t"Homo sapiens"\t"Homo sapiens"\t"Homo
sapiens"\t"Homo sapiens"\t"Homo sapiens"\t"Homo sapiens"\t"Homo
sapiens"\t"Hom'
Line 39: '!Sample_characteristics_ch1\t"group: Control"\t"group:
Control"\t"group: Control"\t"group: Control"\t"group: Control"\
t"group: Control"\t"group: Control"\t"group: Control"\t"group:
Control"\t"group: Control"\t"group: Control"\t"group: Control"\
t"group: Control"\t"'
Line 40: '!Sample_characteristics_ch1\t"time after stroke (h):
Control"\t"time after stroke (h): Control"\t"time after stroke (h):
Control"\t"time after stroke (h): Control"\t"time after stroke (h):
Control"\t"time after stroke (h): Control"\t"time after stroke (h): '
Line 41: '!Sample_characteristics_ch1\t"subject id: Control-015"\
t"subject id: Control-017"\t"subject id: Control-024"\t"subject id:
Control-029"\t"subject id: Control-034"\t"subject id: Control-041"\
t"subject id: Control-050"\t"subject id: Control-057"\t"subject id: C'
Line 42: '!Sample_characteristics_ch1\t"tissue: blood"\t"tissue:
blood"\t"tissue: blood"\t"tissue: blood"\t"tissue: blood"\t"tissue:
blood"\t"tissue: blood"\t"tissue: blood"\t"tissue: blood"\t"tissue:
blood"\t"tissue: blood"\t"tissue: blood"\t"tissue: blood"\t"tissue:
blood'
Line 43: '!Sample_molecule_ch1\t"total RNA"\t"total RNA"\t"total RNA"\
t"total RNA"\t"total RNA"\t"total RNA"\t"total RNA"\t"total RNA"\
t"total RNA"\t"total RNA"\t"total RNA"\t"total RNA"\t"total RNA"\
t"total RNA"\t"total RNA"\t"total RNA"\t"total RNA"\t"total RNA"\
t"total RNA"\t"'
Line 44: '!Sample_extract_protocol_ch1\t"Whole blood was collected
from each subject via antecubital fossa venipuncture into PAXgene
Vacutainer tubes (Qiagen, Valencia, CA, USA). These tubes contain a
solution that immediately lyses all of the cells in whole b'
Line 45: '!Sample_label_ch1\t"biotin"\t"biotin"\t"biotin"\t"biotin"\
t"biotin"\t"biotin"\t"biotin"\t"biotin"\t"biotin"\t"biotin"\t"biotin"\
t"biotin"\t"biotin"\t"biotin"\t"biotin"\t"biotin"\t"biotin"\t"biotin"\
t"biotin"\t"biotin"\t"biotin"\t"biotin"\t"biotin"\t"biotin"\t"biotin"\
t"biotin'

```
Line 46: '!Sample_label_protocol_ch1\t"50\xa0ng RNA samples were
amplified using the RNA Amplification System V2 (Catalogue #3100)
(NuGEN, San Carlos, CA); Ovation WB Reagent (Catalogue #1300); Encore
® Biotin Module (Catalogue #4200) (NuGEN, San Carlos, CA)."\t"50'
Line 47: '!Sample_taxid_ch1\t"9606"\t"9606"\t"9606"\t"9606"\t"9606"\
t"9606"\t"9606"\t"9606"\t"9606"\t"9606"\t"9606"\t"9606"\t"9606"\
t"9606"\t"9606"\t"9606"\t"9606"\t"9606"\t"9606"\t"9606"\t"9606"\
t"9606"\t"9606"\t"9606"\t"9606"\t"9606"\t"9606"\t"9606"\t"9606"\
t"9606"\t"9606"\t"9606"\t"9606"\t"'
Line 48: '!Sample_hyb_protocol\t"Hybridization, washing and laser
scanning of Whole Genome U133 Affymetrix Arrays were performed
according to protocol (Affymetrix, Santa Clara, CA)."\t"Hybridization,
washing and laser scanning of Whole Genome U133 Affymetrix Arr'
Line 49: '!Sample_scan_protocol\t"Hybridization, washing and laser
scanning of Whole Genome U133 Affymetrix Arrays were performed
according to protocol (Affymetrix, Santa Clara, CA)."\t"Hybridization,
washing and laser scanning of Whole Genome U133 Affymetrix Ar'
Line 50: '!Sample_data_processing\t"Raw CEL file data was uploaded in
Partek Genomics Suite and was internal gene normalized (Stamova et al,
BMC Medical Genomics, 2009)."\t"Raw CEL file data was uploaded in
Partek Genomics Suite and was internal gene normalized '
Line 51: '!Sample_platform_id\t"GPL570"\t"GPL570"\t"GPL570"\t"GPL570"\
t"GPL570"\t"GPL570"\t"GPL570"\t"GPL570"\t"GPL570"\t"GPL570"\t"GPL570"\
t"GPL570"\t"GPL570"\t"GPL570"\t"GPL570"\t"GPL570"\t"GPL570"\t"GPL570"\
t"GPL570"\t"GPL570"\t"GPL570"\t"GPL570"\t"GPL570"\t"GPL570"\t"GPL570"\
t"GPL5'
Line 52: '!Sample_contact_name\t"Boryana,,Stamova"\
t"Boryana,,Stamova"\t"Boryana,,Stamova"\t"Boryana,,Stamova"\
t"Boryana,,Stamova"\t"Boryana,,Stamova"\t"Boryana,,Stamova"\
t"Boryana,,Stamova"\t"Boryana,,Stamova"\t"Boryana,,Stamova"\
t"Boryana,,Stamova"\t"Boryana,,Stamova"\t"'
Line 53: '!Sample_contact_email\t"boryana.stamova@ucdmc.ucdavis.edu"\
t"boryana.stamova@ucdmc.ucdavis.edu"\
t"boryana.stamova@ucdmc.ucdavis.edu"\
t"boryana.stamova@ucdmc.ucdavis.edu"\
t"boryana.stamova@ucdmc.ucdavis.edu"\
t"boryana.stamova@ucdmc.ucdavis.edu"\t"boryana.sta'
Line 54: '!Sample_contact_phone\t"916-703-0384"\t"916-703-0384"\t"916-
703-0384"\t"916-703-0384"\t"916-703-0384"\t"916-703-0384"\t"916-703-
0384"\t"916-703-0384"\t"916-703-0384"\t"916-703-0384"\t"916-703-0384"\
t"916-703-0384"\t"916-703-0384"\t"916-703-0384"\t"916-703-0384"\t"91'
Line 55: '!Sample_contact_laboratory\t"Dr. Frank Sharp"\t"Dr. Frank
Sharp"\t"Dr. Frank Sharp"\t"Dr. Frank Sharp"\t"Dr. Frank Sharp"\t"Dr.
Frank Sharp"\t"Dr. Frank Sharp"\t"Dr. Frank Sharp"\t"Dr. Frank Sharp"\
t"Dr. Frank Sharp"\t"Dr. Frank Sharp"\t"Dr. Frank Sharp"\t"Dr. Fr'
Line 56: '!Sample_contact_department\t"Neurology; MIND Institute"\
t"Neurology; MIND Institute"\t"Neurology; MIND Institute"\t"Neurology;
MIND Institute"\t"Neurology; MIND Institute"\t"Neurology; MIND
Institute"\t"Neurology; MIND Institute"\t"Neurology; MIND Institute"'
```

```
Line 57: '!Sample_contact_institute\t"University of California, Davis
Medical Center"\t"University of California, Davis Medical Center"\
t"University of California, Davis Medical Center"\t"University of
California, Davis Medical Center"\t"University of California, D'
Line 58: '!Sample_contact_address\t"2805 50th Street"\t"2805 50th
Street"\t"2805 50th Street"\t"2805 50th Street"\t"2805 50th Street"\
t"2805 50th Street"\t"2805 50th Street"\t"2805 50th Street"\t"2805
50th Street"\t"2805 50th Street"\t"2805 50th Street"\t"2805 50th
Street'
Line 59: '!Sample_contact_city\t"Sacramento"\t"Sacramento"\
t"Sacramento"\t"Sacramento"\t"Sacramento"\t"Sacramento"\t"Sacramento"\
t"Sacramento"\t"Sacramento"\t"Sacramento"\t"Sacramento"\t"Sacramento"\
t"Sacramento"\t"Sacramento"\t"Sacramento"\t"Sacramento"\t"Sacramento"\
t"Sacrame'
Line 60: '!Sample_contact_state\t"California"\t"California"\
t"California"\t"California"\t"California"\t"California"\t"California"\
t"California"\t"California"\t"California"\t"California"\t"California"\
t"California"\t"California"\t"California"\t"California"\t"California"\
t"Califo'
Line 61: '!Sample_contact_zip/postal_code\t"95817"\t"95817"\t"95817"\
t"95817"\t"95817"\t"95817"\t"95817"\t"95817"\t"95817"\t"95817"\t"95817"\
t"95817"\t"95817"\t"95817"\t"95817"\t"95817"\t"95817"\t"95817"\t"95817"\
t"95817"\t"95817"\t"95817"\t"95817"\t"95817"\t"95817"\t"95817"\t"95817"\
t"95817"\t"95817"\t"95817"\t"9'
Line 62: '!Sample_contact_country\t"USA"\t"USA"\t"USA"\t"USA"\t"USA"\
t"USA"\t"USA"\t"USA"\t"USA"\t"USA"\t"USA"\t"USA"\t"USA"\t"USA"\t"USA"\
t"USA"\t"USA"\t"USA"\t"USA"\t"USA"\t"USA"\t"USA"\t"USA"\t"USA"\t"USA"\
t"USA"\t"USA"\t"USA"\t"USA"\t"USA"\t"USA"\t"USA"\t"USA"\t"USA"\t"USA"\
t"USA"\t"USA"\t"USA'
Line 63:
'!Sample_supplementary_file\t"ftp://ftp.ncbi.nlm.nih.gov/geo/samples/
GSM1406nnn/GSM1406033/suppl/GSM1406033_Control-015_HG-
U133_Plus_2_.CEL.gz"\t"ftp://ftp.ncbi.nlm.nih.gov/geo/samples/
GSM1406nnn/GSM1406034/suppl/GSM1406034_Control-017_HG-U133_Plus_2_.C'
Line 64: '!Sample_data_row_count\t"54675"\t"54675"\t"54675"\t"54675"\
t"54675"\t"54675"\t"54675"\t"54675"\t"54675"\t"54675"\t"54675"\
t"54675"\t"54675"\t"54675"\t"54675"\t"54675"\t"54675"\t"54675"\
t"54675"\t"54675"\t"54675"\t"54675"\t"54675"\t"54675"\t"54675"\
t"54675"\t"54675"\t"54675"\t"54'
```

```python
import gzip
import pandas as pd

filepath = "/kaggle/working/GSE58294_series_matrix.txt.gz"

# Extract sample metadata
samples, groups, timepoints = [], [], []

with gzip.open(filepath, "rt") as f:
    for line in f:
```

```
        line = line.strip()
        if line.startswith("!Sample_geo_accession"):
            samples = [x.strip('"') for x in line.split("\t")[1:]]
        elif "group:" in line:
            groups = [x.strip('"').replace("group: ", "") for x in
line.split("\t")[1:]]
        elif "time after stroke" in line:
            timepoints = [x.strip('"').split(": ")[-1] for x in
line.split("\t")[1:]]
        elif line.startswith("!series_matrix_table_begin"):
            break

meta = pd.DataFrame({
    "sample": samples,
    "group": groups,
    "time_after_stroke": timepoints
})

print("Shape:", meta.shape)
print("\nGroup counts:")
print(meta["group"].value_counts())
print("\nTime points:")
print(meta["time_after_stroke"].value_counts())
print("\nFirst 10 rows:")
print(meta.head(10))
```

```
Shape: (92, 3)

Group counts:
group
Cardioembolic Stroke    69
Control                 23
Name: count, dtype: int64

Time points:
time_after_stroke
Control    23
3          23
5          23
24         23
Name: count, dtype: int64

First 10 rows:
       sample    group time_after_stroke
0  GSM1406033  Control           Control
1  GSM1406034  Control           Control
2  GSM1406035  Control           Control
3  GSM1406036  Control           Control
4  GSM1406037  Control           Control
5  GSM1406038  Control           Control
```

```
6  GSM1406039  Control           Control
7  GSM1406040  Control           Control
8  GSM1406041  Control           Control
9  GSM1406042  Control           Control
```

```
import gzip
import pandas as pd

filepath = "/kaggle/working/GSE58294_series_matrix.txt.gz"

# Load expression matrix
skip = 0
with gzip.open(filepath, "rt") as f:
    for i, line in enumerate(f):
        if line.startswith("!series_matrix_table_begin"):
            skip = i + 1
            break

expr = pd.read_csv(filepath, sep="\t", skiprows=skip, index_col=0,
compression="gzip")
expr = expr[expr.index != "!series_matrix_table_end"]

print("Full expression matrix shape:", expr.shape)

# Filter to our NK signature probes
nk_probes = [
    "212843_at",   # GNLY
    "207723_s_at", # NKG7
    "209160_at",   # PRF1
    "202458_at",   # GZMB
    "226279_at",   # KLRD1
    "205495_s_at", # KLRB1
    "207795_s_at", # NCR1
    "226809_at",   # FCGR3A
    "218638_s_at", # SH2D1B
    "227394_at",   # PRSS23
]

probe_to_gene = {
    "212843_at": "GNLY", "207723_s_at": "NKG7", "209160_at": "PRF1",
    "202458_at": "GZMB", "226279_at": "KLRD1", "205495_s_at": "KLRB1",
    "207795_s_at": "NCR1", "226809_at": "FCGR3A", "218638_s_at":
"SH2D1B",
    "227394_at": "PRSS23"
}

nk_expr = expr.loc[expr.index.isin(nk_probes)].T
nk_expr.index.name = "sample"
nk_expr = nk_expr.rename(columns=probe_to_gene)
nk_expr = nk_expr.reset_index()
```

```
# Merge with metadata
df = meta.merge(nk_expr, on="sample")
df["is_stroke"] = (df["group"] == "Cardioembolic Stroke").astype(int)

print("\nFinal dataset shape:", df.shape)
print("\nFirst 5 rows:")
print(df[["sample","group","time_after_stroke","GNLY","NKG7","GZMB","P
RSS23"]].head())
```

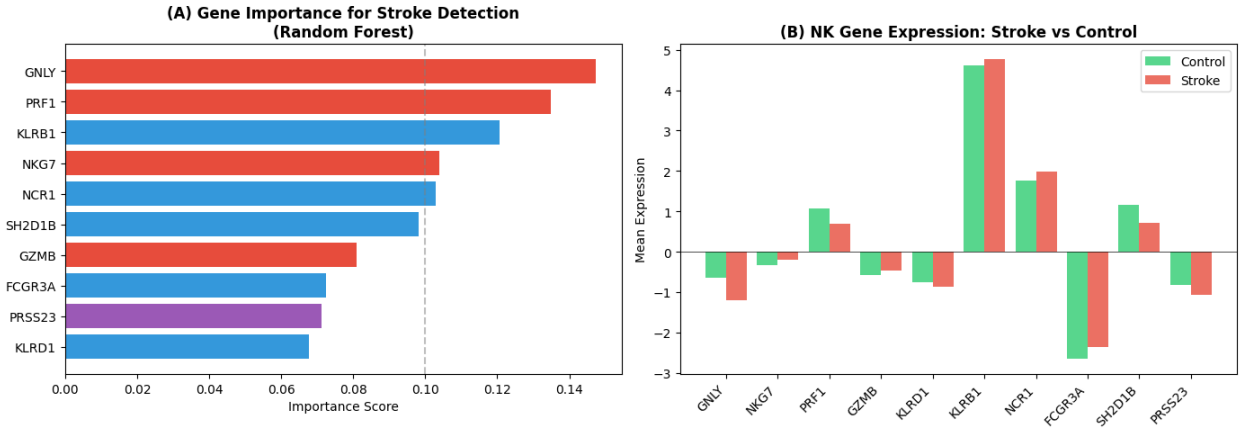
```
Full expression matrix shape: (54675, 92)

Final dataset shape: (92, 14)

First 5 rows:
       sample    group time_after_stroke     GNLY     NKG7     GZMB
PRSS23
0  GSM1406033  Control           Control -0.86418 -1.26614 -0.89041 -
1.05862
1  GSM1406034  Control           Control -0.41293  0.46000 -0.98108 -
0.63933
2  GSM1406035  Control           Control -0.28249 -0.14155 -1.25024 -
1.18289
3  GSM1406036  Control           Control -0.01084 -0.70668 -0.70668 -
0.86098
4  GSM1406037  Control           Control -1.19363 -0.06484 -1.72623 -
1.81479
```

```
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score, StratifiedKFold
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")

gene_cols =
["GNLY","NKG7","PRF1","GZMB","KLRD1","KLRB1","NCR1","FCGR3A","SH2D1B",
"PRSS23"]
X = df[gene_cols].values
y = df["is_stroke"].values

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

rf = RandomForestClassifier(n_estimators=200, random_state=42)
```

```
lr = LogisticRegression(random_state=42, max_iter=1000)

rf_acc  = cross_val_score(rf, X_scaled, y, cv=cv, scoring="accuracy")
lr_acc  = cross_val_score(lr, X_scaled, y, cv=cv, scoring="accuracy")
rf_auc  = cross_val_score(rf, X_scaled, y, cv=cv, scoring="roc_auc")
lr_auc  = cross_val_score(lr, X_scaled, y, cv=cv, scoring="roc_auc")

print("=" * 45)
print("       STROKE vs CONTROL CLASSIFIER")
print("=" * 45)
print(f"Random Forest  accuracy: {rf_acc.mean():.1%} ±
{rf_acc.std():.1%}")
print(f"Random Forest  AUC:      {rf_auc.mean():.3f} ±
{rf_auc.std():.3f}")
print(f"Logistic Reg   accuracy: {lr_acc.mean():.1%} ±
{lr_acc.std():.1%}")
print(f"Logistic Reg   AUC:      {lr_auc.mean():.3f} ±
{lr_auc.std():.3f}")

# Feature importance
rf.fit(X_scaled, y)
importances = pd.Series(rf.feature_importances_,
index=gene_cols).sort_values(ascending=True)

print("\n=== GENE IMPORTANCE FOR STROKE DETECTION ===")
for gene, imp in importances.items():
    bar = "" * int(imp * 100)
    print(f"  {gene:<8} {imp:.3f}  {bar}")

# Plot
fig, axes = plt.subplots(1, 2, figsize=(14, 5))

# Feature importance bar chart
colors = ["#9b59b6" if g == "PRSS23" else "#e74c3c" if g in
["GNLY","NKG7","GZMB","PRF1"] else "#3498db" for g in
importances.index]
axes[0].barh(importances.index, importances.values, color=colors)
axes[0].set_title("(A) Gene Importance for Stroke Detection\n(Random
Forest)", fontweight="bold")
axes[0].set_xlabel("Importance Score")
axes[0].axvline(x=0.1, color="gray", linestyle="--", alpha=0.5)

# Gene expression stroke vs control
stroke_mean = df[df.is_stroke==1][gene_cols].mean()
control_mean = df[df.is_stroke==0][gene_cols].mean()
x = np.arange(len(gene_cols))
axes[1].bar(x - 0.2, control_mean, 0.4, label="Control",
color="#2ecc71", alpha=0.8)
axes[1].bar(x + 0.2, stroke_mean, 0.4, label="Stroke",
color="#e74c3c", alpha=0.8)
```

```
axes[1].set_xticks(x)
axes[1].set_xticklabels(gene_cols, rotation=45, ha="right")
axes[1].set_title("(B) NK Gene Expression: Stroke vs Control",
fontweight="bold")
axes[1].set_ylabel("Mean Expression")
axes[1].legend()
axes[1].axhline(y=0, color="black", linewidth=0.5)

plt.tight_layout()
plt.savefig("stroke_classifier_results.png", dpi=150)
plt.show()
print("\nFigure saved!")
```

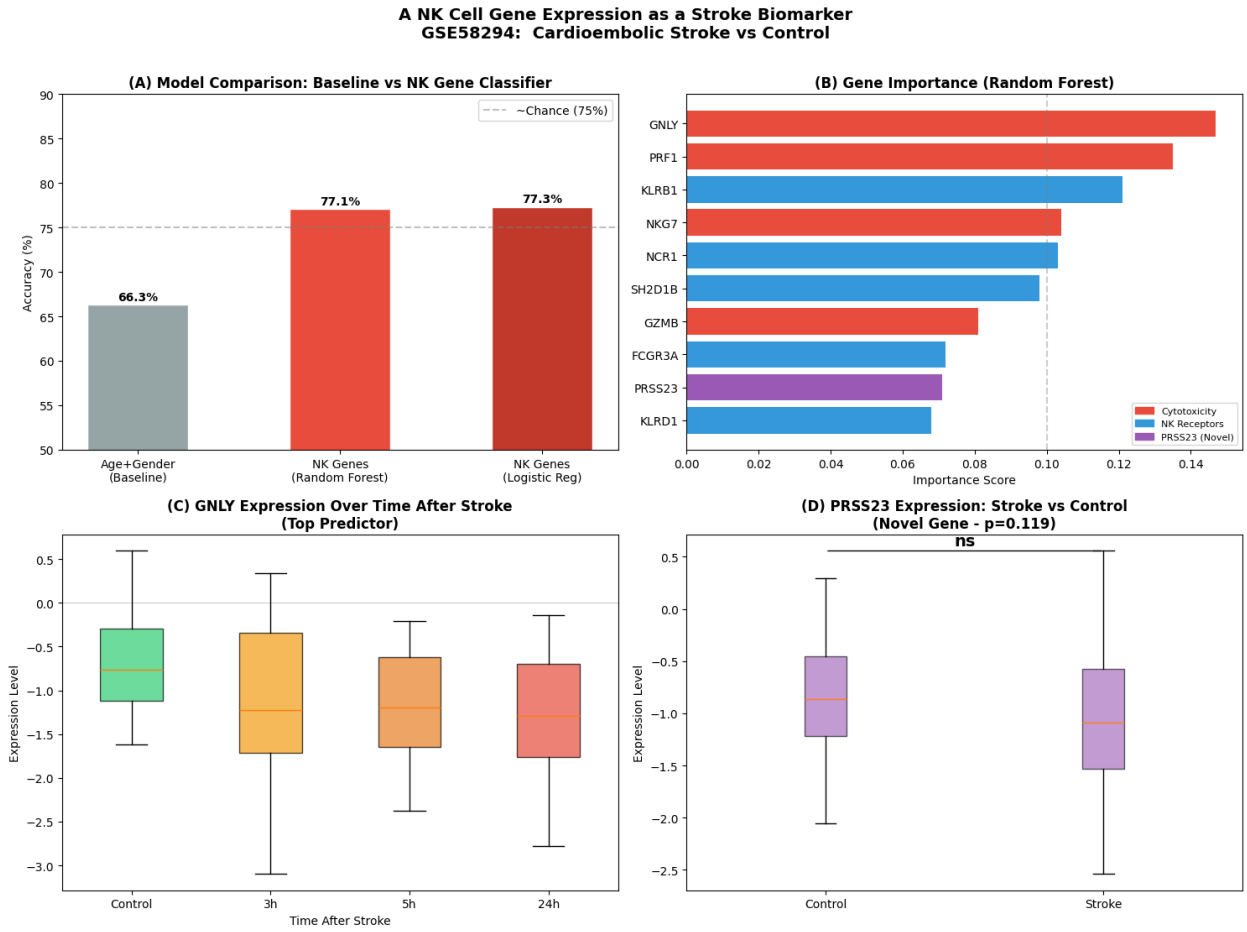
```
============================================
       STROKE vs CONTROL CLASSIFIER
============================================
Random Forest  accuracy: 77.1% ± 5.7%
Random Forest  AUC:      0.692 ± 0.098
Logistic Reg   accuracy: 77.3% ± 5.0%
Logistic Reg   AUC:      0.798 ± 0.048

=== GENE IMPORTANCE FOR STROKE DETECTION ===
  KLRD1    0.068
  PRSS23   0.071
  FCGR3A   0.072
  GZMB     0.081
  SH2D1B   0.098
  NCR1     0.103
  NKG7     0.104
  KLRB1    0.121
  PRF1     0.135
  GNLY     0.147
```




```
Figure saved!
```

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from scipy import stats

gene_cols =
["GNLY","NKG7","PRF1","GZMB","KLRD1","KLRB1","NCR1","FCGR3A","SH2D1B",
"PRSS23"]

fig, axes = plt.subplots(2, 2, figsize=(15, 11))
fig.suptitle("A NK Cell Gene Expression as a Stroke Biomarker\
nGSE58294:  Cardioembolic Stroke vs Control",
             fontsize=14, fontweight="bold", y=1.01)

# Plot 1 — Classifier comparison
models = ["Age+Gender\n(Baseline)", "NK Genes\n(Random Forest)", "NK
Genes\n(Logistic Reg)"]
accs   = [66.3, 77.1, 77.3]
aucs   = [None, 0.692, 0.798]
colors = ["#95a5a6", "#e74c3c", "#c0392b"]
bars = axes[0,0].bar(models, accs, color=colors, edgecolor="white",
width=0.5)
axes[0,0].set_ylim(50, 90)
axes[0,0].set_ylabel("Accuracy (%)")
axes[0,0].set_title("(A) Model Comparison: Baseline vs NK Gene
Classifier", fontweight="bold")
axes[0,0].axhline(y=75, color="gray", linestyle="--", alpha=0.5,
label="~Chance (75%)")
for bar, acc in zip(bars, accs):
    axes[0,0].text(bar.get_x() + bar.get_width()/2, bar.get_height() +
0.5,
                   f"{acc}%", ha="center", fontweight="bold")
axes[0,0].legend()

# Plot 2 — Top gene importances
importances = {
    "GNLY":0.147,"PRF1":0.135,"KLRB1":0.121,"NKG7":0.104,
    "NCR1":0.103,"SH2D1B":0.098,"GZMB":0.081,"FCGR3A":0.072,
    "PRSS23":0.071,"KLRD1":0.068
}
imp_series = pd.Series(importances).sort_values()
bar_colors = ["#9b59b6" if g=="PRSS23" else "#e74c3c" if g in
["GNLY","PRF1","GZMB","NKG7"] else "#3498db"
              for g in imp_series.index]
axes[0,1].barh(imp_series.index, imp_series.values, color=bar_colors)
axes[0,1].set_title("(B) Gene Importance (Random Forest)",
fontweight="bold")
axes[0,1].set_xlabel("Importance Score")
axes[0,1].axvline(x=0.1, color="gray", linestyle="--", alpha=0.4)
from matplotlib.patches import Patch
```

```
legend_els = [Patch(color="#e74c3c",label="Cytotoxicity"),
              Patch(color="#3498db",label="NK Receptors"),
              Patch(color="#9b59b6",label="PRSS23 (Novel)")]
axes[0,1].legend(handles=legend_els, fontsize=8)

# Plot 3 — GNLY across timepoints (top gene)
timepoint_order = ["Control","3","5","24"]
gnly_by_time = [df[df.time_after_stroke==t]["GNLY"].values for t in
timepoint_order]
bp = axes[1,0].boxplot(gnly_by_time,
labels=["Control","3h","5h","24h"],
                       patch_artist=True, notch=False)
tp_colors = ["#2ecc71","#f39c12","#e67e22","#e74c3c"]
for patch, color in zip(bp["boxes"], tp_colors):
    patch.set_facecolor(color)
    patch.set_alpha(0.7)
axes[1,0].set_title("(C) GNLY Expression Over Time After Stroke\n(Top
Predictor)", fontweight="bold")
axes[1,0].set_xlabel("Time After Stroke")
axes[1,0].set_ylabel("Expression Level")
axes[1,0].axhline(y=0, color="black", linewidth=0.5, alpha=0.3)

# Plot 4 — PRSS23 stroke vs control with stats
prss23_control = df[df.is_stroke==0]["PRSS23"].values
prss23_stroke  = df[df.is_stroke==1]["PRSS23"].values
t_stat, p_val = stats.ttest_ind(prss23_control, prss23_stroke)
axes[1,1].boxplot([prss23_control, prss23_stroke],
                  labels=["Control","Stroke"],
                  patch_artist=True,
                  boxprops=dict(facecolor="#9b59b6", alpha=0.6))
axes[1,1].set_title(f"(D) PRSS23 Expression: Stroke vs Control\n(Novel
Gene - p={p_val:.3f})", fontweight="bold")
axes[1,1].set_ylabel("Expression Level")
sig = "*" if p_val < 0.05 else "ns"
y_max = max(max(prss23_control), max(prss23_stroke))
axes[1,1].annotate("", xy=(2, y_max), xytext=(1, y_max),
                   arrowprops=dict(arrowstyle="-", color="black"))
axes[1,1].text(1.5, y_max+0.05, sig, ha="center", fontsize=14,
fontweight="bold")

plt.tight_layout()
plt.savefig("full_results_figure.png", dpi=150, bbox_inches="tight")
plt.show()
print("Figure saved!")
print(f"\nPRSS23 t-test: t={t_stat:.3f}, p={p_val:.4f}")
print(f"Control mean: {prss23_control.mean():.3f}")
print(f"Stroke mean:  {prss23_stroke.mean():.3f}")
```




```
Figure saved!

PRSS23 t-test: t=1.572, p=0.1195
Control mean: -0.834
Stroke mean:  -1.069
```

```python
import pandas as pd
import numpy as np
from scipy import stats

gene_cols =
["GNLY","NKG7","PRF1","GZMB","KLRD1","KLRB1","NCR1","FCGR3A","SH2D1B",
"PRSS23"]

control = df[df.is_stroke==0]
stroke  = df[df.is_stroke==1]

print("=" * 65)
print(f"{'Gene':<10} {'Control Mean':>13} {'Stroke Mean':>12} {'p-
value':>10} {'Sig':>5}")
print("=" * 65)
```

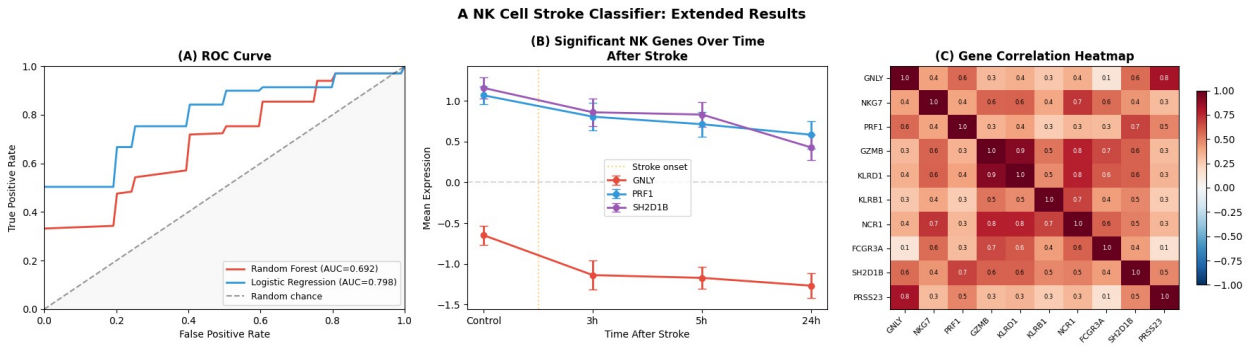
```
results = []
for gene in gene_cols:
    c = control[gene].values
    s = stroke[gene].values
    t, p = stats.ttest_ind(c, s)
    sig = "***" if p < 0.001 else "**" if p < 0.01 else "*" if p <
0.05 else "ns"
    print(f"{gene:<10} {c.mean():>13.3f} {s.mean():>12.3f} {p:>10.4f}
{sig:>5}")
    results.append({"gene": gene, "control_mean": c.mean(),
"stroke_mean": s.mean(), "p_value": p, "sig": sig})

results_df = pd.DataFrame(results)
print("=" * 65)
print("\n* p<0.05  ** p<0.01  *** p<0.001  ns=not significant")
```

```
=================================================================
Gene        Control Mean  Stroke Mean    p-value   Sig
=================================================================
GNLY              -0.650       -1.194     0.0017    **
NKG7              -0.327       -0.209     0.5950    ns
PRF1               1.069        0.702     0.0381     *
GZMB              -0.573       -0.461     0.6204    ns
KLRD1             -0.749       -0.876     0.6228    ns
KLRB1              4.620        4.777     0.1741    ns
NCR1               1.769        1.991     0.1855    ns
FCGR3A            -2.660       -2.356     0.1781    ns
SH2D1B             1.160        0.707     0.0133     *
PRSS23            -0.834       -1.069     0.1195    ns
=================================================================

* p<0.05  ** p<0.01  *** p<0.001  ns=not significant
```

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import StratifiedKFold
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import roc_curve, auc
from scipy import stats

gene_cols =
["GNLY","NKG7","PRF1","GZMB","KLRD1","KLRB1","NCR1","FCGR3A","SH2D1B",
"PRSS23"]
X = df[gene_cols].values
y = df["is_stroke"].values
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
fig, axes = plt.subplots(1, 3, figsize=(18, 5))
fig.suptitle("A NK Cell Stroke Classifier: Extended Results",
fontsize=14, fontweight="bold")

# ── Plot 1: ROC Curve ──────────────────────────────────────
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
rf = RandomForestClassifier(n_estimators=200, random_state=42)
lr = LogisticRegression(random_state=42, max_iter=1000)

for model, name, color in [(rf, "Random Forest", "#e74c3c"), (lr,
"Logistic Regression", "#3498db")]:
    tprs, aucs_list, mean_fpr = [], [], np.linspace(0, 1, 100)
    for train, test in cv.split(X_scaled, y):
        model.fit(X_scaled[train], y[train])
        probs = model.predict_proba(X_scaled[test])[:, 1]
        fpr, tpr, _ = roc_curve(y[test], probs)
        tprs.append(np.interp(mean_fpr, fpr, tpr))
        aucs_list.append(auc(fpr, tpr))
    mean_tpr = np.mean(tprs, axis=0)
    mean_auc = np.mean(aucs_list)
    axes[0].plot(mean_fpr, mean_tpr, color=color, label=f"{name}
(AUC={mean_auc:.3f})", linewidth=2)

axes[0].plot([0,1],[0,1],"k--", alpha=0.4, label="Random chance")
axes[0].fill_between([0,1],[0,1], alpha=0.05, color="gray")
axes[0].set_xlabel("False Positive Rate")
axes[0].set_ylabel("True Positive Rate")
axes[0].set_title("(A) ROC Curve", fontweight="bold")
axes[0].legend(loc="lower right", fontsize=9)
axes[0].set_xlim([0,1])
axes[0].set_ylim([0,1])

# ── Plot 2: Top 3 significant genes over time ──────────────
timepoints = ["Control", "3", "5", "24"]
labels     = ["Control", "3h", "5h", "24h"]
sig_genes  = ["GNLY", "PRF1", "SH2D1B"]
colors_tp  = {"GNLY": "#e74c3c", "PRF1": "#3498db", "SH2D1B":
"#9b59b6"}

for gene in sig_genes:
    means = [df[df.time_after_stroke==t][gene].mean() for t in
timepoints]
    sems  = [df[df.time_after_stroke==t][gene].sem()  for t in
timepoints]
    axes[1].errorbar(labels, means, yerr=sems, marker="o",
linewidth=2,
                     capsize=4, label=gene, color=colors_tp[gene])

axes[1].axhline(y=0, color="gray", linestyle="--", alpha=0.3)
```

```
axes[1].axvline(x=0.5, color="orange", linestyle=":", alpha=0.5,
label="Stroke onset")
axes[1].set_title("(B) Significant NK Genes Over Time\nAfter Stroke",
fontweight="bold")
axes[1].set_xlabel("Time After Stroke")
axes[1].set_ylabel("Mean Expression")
axes[1].legend(fontsize=9)

# ── Plot 3: Heatmap of gene correlations ──────────────────────
corr = df[gene_cols].corr()
im = axes[2].imshow(corr, cmap="RdBu_r", vmin=-1, vmax=1,
aspect="auto")
axes[2].set_xticks(range(len(gene_cols)))
axes[2].set_yticks(range(len(gene_cols)))
axes[2].set_xticklabels(gene_cols, rotation=45, ha="right",
fontsize=8)
axes[2].set_yticklabels(gene_cols, fontsize=8)
axes[2].set_title("(C) Gene Correlation Heatmap", fontweight="bold")
plt.colorbar(im, ax=axes[2], shrink=0.8)
for i in range(len(gene_cols)):
    for j in range(len(gene_cols)):
        axes[2].text(j, i, f"{corr.iloc[i,j]:.1f}",
                     ha="center", va="center", fontsize=6,
                     color="white" if abs(corr.iloc[i,j]) > 0.6 else
"black")

plt.tight_layout()
plt.savefig("extended_results.png", dpi=150, bbox_inches="tight")
plt.show()

# ── Timepoint stats ──────────────────────────────────────────
print("\n=== GENE EXPRESSION BY TIMEPOINT ===")
print(f"{'Gene':<10} {'Control':>9} {'3h':>9} {'5h':>9} {'24h':>9}
{'Trend'}")
print("-" * 55)
for gene in sig_genes:
    vals = [df[df.time_after_stroke==t][gene].mean() for t in
timepoints]
    trend = "↓ drops" if vals[-1] < vals[1] else "↑ rises"
    print(f"{gene:<10} {vals[0]:>9.3f} {vals[1]:>9.3f} {vals[2]:>9.3f}
{vals[3]:>9.3f}  {trend}")
```




```
=== GENE EXPRESSION BY TIMEPOINT ===
Gene          Control        3h        5h       24h Trend
------------------------------------------------------------
GNLY           -0.650    -1.140    -1.173    -1.269  ↓ drops
PRF1            1.069     0.807     0.714     0.585  ↓ drops
SH2D1B          1.160     0.861     0.833     0.428  ↓ drops
```

```python
from scipy import stats
import pandas as pd

sig_genes = ["GNLY", "PRF1", "SH2D1B"]
timepoints = ["3", "5", "24"]

print("=== TIMEPOINT vs CONTROL p-values ===\n")
print(f"{'Gene':<10} {'vs 3h':>10} {'vs 5h':>10} {'vs 24h':>10}")
print("-" * 45)

control = df[df.time_after_stroke=="Control"]

for gene in sig_genes:
    pvals = []
    for t in timepoints:
        group = df[df.time_after_stroke==t][gene].values
        _, p = stats.ttest_ind(control[gene].values, group)
        pvals.append(f"{p:.4f}")
    print(f"{gene:<10} {pvals[0]:>10} {pvals[1]:>10} {pvals[2]:>10}")

print("\n# values under 0.05 are significant")
```

```
=== TIMEPOINT vs CONTROL p-values ===

Gene             vs 3h      vs 5h     vs 24h
---------------------------------------------
GNLY           0.0263     0.0052     0.0025
PRF1           0.1984     0.0663     0.0198
SH2D1B         0.1672     0.1121     0.0009

# values under 0.05 are significant
```