

Pre-training Assessment of Model Transferability via Distribution Shift in Petroleum Engineering

Javad Afshari

J.Afshari13610220@gmail.com

Islamic Azad University of Birjand <https://orcid.org/0009-0001-0199-9202>

Mahmoud Afshari

Birjand University of Technology <https://orcid.org/0000-0002-8556-4264>

Hossien Afshari

University of Birjand <https://orcid.org/0000-0003-4981-2504>

Research Article

Keywords: Data-driven modeling, Distribution shift, Model transferability, Pre-training assessment, Petroleum engineering, Statistical distance, Machine learning

Posted Date: October 27th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-11502138/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.
[Read Full License](#)

Additional Declarations: The authors declare no competing interests.

Pre-training Assessment of Model Transferability via Distribution Shift in Petroleum Engineering

Javad Afshari^{1*}, Mahmoud Afshari², Hossein Afshari³

¹ Department of Computer Engineering, Islamic Azad University of Birjand, Birjand, Iran

² Department of Mechanical Engineering, Birjand University of Technology, Birjand, Iran

³ Department of Mechanical Engineering, University of Birjand, Birjand, Iran

* Corresponding author's email: J.Afshari13610220@gmail.com

Abstract— Data-driven models are increasingly used in petroleum engineering for tasks such as reservoir characterization, well-log interpretation, and production prediction. However, predictive performance may change when a model is applied to data from a different well, field, or operating condition. Differences between source and target datasets can therefore limit model transferability, yet this issue is commonly assessed only after model training. This study proposes a pre-training framework for quantifying source–target distribution shift and examining its relationship with subsequent zero-shot model transfer performance. The proposed approach uses predefined engineering features and Wasserstein-based distributional distances to construct a model-independent shift index that can be calculated before predictive model development. A prediction model is then trained exclusively on the source data and evaluated on the target data without retraining or adaptation. Pearson and Spearman correlation analyses are used to examine the relationship between the pre-training shift index and relative performance degradation. A controlled synthetic experiment is first used to verify the computational pipeline and methodological design. Across 24 synthetic well-transfer cases, the proposed indices captured different levels of source–target distribution shift, while the zero-shot model performance varied across target wells. None of the tested correlations reached the conventional 0.05 significance level, indicating that the present experiment provides preliminary rather than conclusive evidence of a systematic relationship between distribution shift and transfer degradation. Application to the real FORCE 2020 dataset is proposed as the next stage of validation. The framework provides a model-independent diagnostic that can be used to identify potentially challenging transfer settings before investing in model development or deployment.

Keywords: *Data-driven modeling; Distribution shift; Model transferability; Pre-training assessment; Petroleum engineering; Statistical distance; Machine learning.*

1 INTRODUCTION

Data-driven and machine learning (ML) methods are now widely used in petroleum engineering. They have been applied to problems such as reservoir characterization, recovery-factor estimation, well-log interpretation, production prediction, and reservoir optimization [1–3]. These models can learn complex relationships from geological, petrophysical, and production data and can provide useful predictions with lower computational cost in some applications.

However, the performance of an ML model can change when it is applied to data from a different well, reservoir, or field. Petroleum datasets are often heterogeneous because geological properties, operating conditions, measurement systems, and data ranges can vary between locations. As a result, the source and target datasets may have different statistical

distributions even when they describe the same engineering problem. Recent studies provide direct evidence of this problem. Roustazadeh et al. [1] showed that the accuracy and reliability of ML models for hydrocarbon recovery-factor estimation depended on the database used for model development. Their results showed that significant differences between databases could lead to unsatisfactory predictions on unseen data. Xu et al. [2] also identified a mismatch between training and testing domains as an important challenge in well-log formation evaluation and used sample similarity to improve model transfer between wells. However, their results showed that the benefit of similarity-based transfer can be limited when the source and target wells are substantially different. Transfer learning is one way to address this problem.

Recent petroleum studies have used transfer learning to adapt models to new data or reduce the

amount of target-domain data required for model development [2,3]. For example, Cui et al. [3] used transfer learning and multi-fidelity data to build surrogate models for reservoir production optimization. These approaches can improve model adaptation, but they mainly focus on modifying or retraining the model after a source–target difference has already been identified. This problem can be viewed more generally as distribution shift.

Distribution shift occurs when the statistical distribution of the target data differs from that of the data used to train the model. Such a shift can reduce predictive performance after deployment. Recent studies have shown that changes in data distribution can be directly related to changes in model performance and can be used to diagnose the sources of performance degradation [4,5].

Despite these developments, an important question remains for petroleum applications: Can the degree of source–target distribution shift be measured before model training, and can this measure provide information about later model transferability? Most existing petroleum studies evaluate transferability through model performance, transfer learning, or similarity-based training strategies [1–3]. These approaches are useful, but they do not provide a simple, model-independent measure that can be calculated before developing a prediction model.

This study addresses this gap by proposing a pre-training assessment of model transferability based on distribution shift. The proposed framework first extracts predefined engineering features from the source and target datasets. It then measures their distributional difference using Wasserstein-based statistical distances. The resulting value is used as a model-independent distribution-shift index.

The study then examines whether this index is related to the loss of predictive performance after model transfer. A model is trained using the source dataset and evaluated on different target datasets without retraining. The distribution-shift index is compared with the observed change in model performance. This provides a direct way to study whether a larger difference between source and target data is associated with a larger degradation in prediction accuracy.

The main contribution of this study is therefore a simple statistical step that can be performed before model training. Rather than replacing existing ML or transfer-learning methods, the proposed framework provides an early assessment of the data conditions under which model transfer may become difficult. This can support more reliable model development and validation across different petroleum datasets.

2 STATISTICAL BASIS OF DISTRIBUTION SHIFT

Let $P_S(X)$ and $P_T(X)$ denote the probability distributions of the source and target datasets, respectively. In an ideal transfer setting, these two distributions should be similar. In practice, however,

changes in geological conditions, operating states, measurement systems, or data collection procedures can lead to

$$P_S(X) \neq P_T(X).$$

This difference is commonly referred to as distribution shift. In domain adaptation theory, the difference between source and target distributions is an important factor in determining the expected target-domain error [6]. Therefore, measuring the distance between P_S and P_T can provide useful information about the difficulty of transferring a model from one domain to another.

For the present study, the distribution shift is considered at the feature level. Let

$$X = [x_1, x_2, \dots, x_d]$$

represent a d -dimensional engineering feature vector. For each feature x_j , the source and target datasets produce two empirical distributions, denoted by P_S^j and P_T^j . The distribution difference can then be measured independently for each feature.

2.1. Wasserstein Distance

The Wasserstein distance provides a natural measure for comparing two probability distributions. For two one-dimensional distributions P and Q , the first-order Wasserstein distance is defined as

$$W_1(P, Q) = \inf_{\gamma \in \Gamma(P, Q)} \int_{\mathbb{R} \times \mathbb{R}} |x - y| d\gamma(x, y),$$

where $\Gamma(P, Q)$ is the set of all joint distributions with marginal distributions P and Q .

For one-dimensional empirical data, the distance can also be written in terms of the cumulative distribution functions:

$$W_1(P, Q) = \int_0^1 |F_P^{-1}(u) - F_Q^{-1}(u)| du.$$

This formulation compares the quantiles of the two distributions and therefore captures both the location and spread of the data. Unlike measures that only compare probabilities at fixed intervals, Wasserstein distance also considers the magnitude of the movement between the two distributions [7].

For two empirical samples,

$$S = \{s_1, s_2, \dots, s_n\}$$

and

$$T = \{t_1, t_2, \dots, t_m\},$$

the empirical Wasserstein distance can be calculated directly from the observations. This property makes the measure suitable for a model-independent pre-training analysis because no prediction model or learned representation is required.

2.2. Multivariate Distribution Shift

Petroleum engineering data are usually multivariate. Therefore, measuring each feature

separately does not fully describe the joint distribution of the data. A direct multivariate Wasserstein calculation can become computationally demanding as the feature dimension increases.

To address this issue, this study uses the Sliced Wasserstein distance. The main idea is to project the multidimensional data onto a set of one-dimensional directions and calculate the Wasserstein distance for each projection. The resulting distances are then aggregated across the projections [8].

Let θ_k denote the k -th unit projection direction. The projected source and target data are

$$X_S^{(k)} = \theta_k^T X_S$$

and

$$X_T^{(k)} = \theta_k^T X_T.$$

The Sliced Wasserstein distance can then be expressed as

$$SW_1(P_S, P_T) = \frac{1}{K} \sum_{k=1}^K W_1(P_S^{(k)}, P_T^{(k)}),$$

where K is the number of projection directions.

This approach converts a multivariate comparison into a set of one-dimensional comparisons. It is therefore suitable for a practical pre-training assessment when the number of engineering features is moderate. Sliced Wasserstein distance has been used to compare high-dimensional distributions and provides a computationally practical alternative to directly solving the full optimal-transport problem [8].

2.3. From Distribution Distance to a Transferability Index

The statistical distance itself is not the final objective of this study. The main goal is to determine whether the measured source–target difference is related to model transferability.

For this purpose, a distribution-shift index $D_{S,T}$ is defined from the distances between the source and target feature distributions. For a set of d features, a general form is

$$D_{S,T} = \frac{1}{d} \sum_{j=1}^d W_1(P_S^j, P_T^j).$$

When the joint feature distribution is considered, the Sliced Wasserstein distance can be used instead:

$$D_{S,T} = SW_1(P_S, P_T).$$

A larger value of $D_{S,T}$ indicates a larger statistical difference between the source and target datasets. Importantly, this index is calculated without using the prediction model. It can therefore be obtained before model training.

The proposed framework then separates two quantities that are often combined in transfer-learning studies. The first is the data-level distribution shift,

measured by $D_{S,T}$. The second is the model-level performance degradation, measured after the source-trained model is applied to the target dataset. Their relationship can be studied using statistical analysis.

For a performance metric M , the relative performance degradation can be defined as

$$\Delta M = \frac{M_S - M_T}{M_S},$$

where M_S is the performance measured on the source setting and M_T is the performance obtained on the target dataset. The central empirical question of this study is whether ΔM changes systematically with $D_{S,T}$.

This formulation provides a direct link between the statistical properties of the data and the observed behavior of the prediction model. It also allows different source–target pairs to be compared using the same model-independent measure.

3 PROPOSED PRE-TRAINING TRANSFERABILITY FRAMEWORK

The proposed framework aims to assess model transferability before training the prediction model. The main idea is to measure the difference between source and target data first and then examine whether this difference is related to the performance loss after model transfer.

The framework has four main steps: (1) extraction of predefined engineering features, (2) measurement of source–target distribution shift, (3) zero-shot transfer of a source-trained model, and (4) statistical analysis of the relationship between distribution shift and performance degradation. The first two steps are performed before model training, while the last two steps are used to evaluate the practical effect of the measured shift.

3.1. Problem Formulation

Let D_S denote the source dataset and D_T a target dataset. The prediction model is trained only on D_S and is then applied to D_T without retraining or target-domain adaptation. This setting allows the effect of the source–target difference to be examined directly.

The proposed method focuses on the following question: Can the difference between source and target data, measured before model training, provide useful information about the later transfer performance?

To answer this question, the framework separates the problem into two quantities. The first is the **data-level** distribution shift, measured before training. The second is the **model-level** performance degradation, measured after transfer.

3.2. Engineering Feature Representation

The distribution-shift assessment uses a fixed set of predefined engineering features. These features are selected based on their physical meaning and their ability to represent the main characteristics of the data.

For each observation, the selected features form a vector:

$$X = [x_1, x_2, \dots, x_d]$$

The same feature extraction procedure is applied to both source and target datasets. Therefore, differences between the resulting distributions reflect changes in the data rather than changes in the feature representation.

Unlike a learned representation, the proposed feature space does not depend on a trained prediction model. This makes the distribution-shift measurement possible before model development.

3.3. Distribution-Shift Index

For each feature, the source and target empirical distributions are compared using the first-order Wasserstein distance [7]. The resulting feature-level distances are combined to obtain a single measure of marginal distribution shift:

$$D_{S,T} = \frac{1}{d} \sum_{j=1}^d W_1(P_S^j, P_T^j)$$

where d is the number of selected features, and P_S^j and P_T^j are the source and target distributions of feature j .

The Sliced Wasserstein distance can also be used when the joint structure of the feature vector needs to be considered [8]. It provides a complementary multivariate measure rather than an alternative definition of the feature-wise index.

A larger value of $D_{S,T}$ indicates a greater statistical difference between the source and target datasets. Since the index does not use the prediction model or its outputs, it can be calculated before model training.

3.4. Zero-shot Model Transfer

After calculating the distribution-shift index, a prediction model is trained using only the source dataset. The trained model is then directly evaluated on the target dataset without retraining or target-domain adaptation. This creates a zero-shot transfer setting.

This design is important because adaptation methods can reduce the effect of distribution shift during training. In contrast, the present study aims to observe the relationship between the initial data difference and the performance loss under direct transfer.

Let M_S represent the model performance on the source dataset and M_T the performance on the target dataset. The relative performance degradation is defined as

$$\Delta M = \frac{M_S - M_T}{M_S}$$

where a larger positive value represents a greater loss of performance for metrics in which higher values indicate better performance.

3.5. Statistical Relationship Analysis

The final step is to examine whether the pre-training distribution-shift index is related to the observed performance degradation. Each source–target pair provides one value of $D_{S,T}$ and one corresponding value of ΔM .

The main hypothesis is that a larger source–target distribution difference may be associated with greater performance degradation after transfer. This relationship is examined using statistical correlation and, where appropriate, regression analysis.

The purpose of this analysis is not to claim that distribution shift is the only cause of model degradation. Instead, it tests whether the proposed model-independent index contains useful information about transfer behavior.

Overall, the framework establishes a simple link between data-level distribution shift and model-level transfer performance. The distribution-shift index is obtained before model training, while the transfer experiment provides the subsequent empirical validation of its usefulness.

4 EXPERIMENTAL SETUP

4.1 Dataset and Data Access

The proposed pre-training transferability assessment was evaluated using the publicly available FORCE 2020 well-log and lithofacies dataset. The dataset contains well-log data from 118 wells in the Norwegian Sea and includes expert-interpreted lithology information for the labeled wells [9]. The complete dataset is publicly available through Zenodo, including the original LAS files and associated metadata. [9]

The FORCE 2020 competition data are organized into 98 training wells, 10 validation wells, and 10 blind test wells. The training set contains 1,170,511 samples, the validation set contains 136,786 samples from 10 wells, and the blind test set contains 122,397 samples from 10 wells [11].

The blind test wells are not used for the main analysis because their reference lithology labels are not available. The 98 labeled training wells are used to construct the primary source–target transfer cases, while the 10 validation wells are retained for an external evaluation of the proposed procedure.

The dataset is obtained directly from the public Zenodo record associated with DOI 10.5281/zenodo.4351156. The downloaded archive contains the original LAS files. In the present study, the dataset is not assumed to be preloaded or manually converted into a separate processed table. Instead, the proposed implementation automatically downloads the public archive, extracts the LAS files, identifies the available wells, reads the logging curves, and constructs the analysis tables.

The complete data-acquisition and LAS-loading procedure is provided in Appendix A, while the complete executable analysis code is provided in Appendix B. Thus, the reported results can be reproduced starting from the publicly accessible FORCE 2020 dataset.

Seven well-log variables are selected as the predefined engineering feature representation:

- gamma ray (GR),
- bulk density (RHOB),
- neutron porosity (NPHI),
- photoelectric factor (PEF),
- deep resistivity (RDEP),
- compressional sonic transit time (DTC), and
- caliper (CALI).

These variables represent different physical characteristics of the formation and provide a compact multivariate representation of the source and target domains.

4.2 Source–Target Transfer Design

Each well is treated as a separate data domain. The primary experiment uses a leave-one-well-out design over the 98 labeled training wells. In each case, one well is defined as the target domain and all remaining labeled wells are used as the source domain.

For a target well T_i , the source domain is therefore defined as

$$S_i = \{W_1, \dots, W_{i-1}, W_{i+1}, \dots, W_{98}\}.$$

The distribution-shift index is calculated between S_i and T_i before the predictive model is trained. The model is subsequently trained only using S_i and is directly evaluated on T_i .

This design produces multiple source–target transfer cases and allows the proposed index to be compared with the observed change in predictive performance across wells.

An additional external experiment is performed using the 10 official validation wells. In this experiment, the 98 training wells are used as the source domain and each validation well is treated as an unseen target domain. The target wells are not used for model training, feature selection, hyperparameter tuning, or normalization.

This experimental setting is related to previous petroleum studies that have examined transfer learning and cross-domain generalization in well-log data. Xu et al. [12], for example, developed a similarity-weighted transfer-learning framework for well-log formation evaluation and explicitly considered differences between training and testing logging data. Zhang et al. [13] investigated transfer learning for LWD gamma-ray depth-lag remediation between the Volve field and a different gas field.

In contrast to these model-adaptation approaches, the present study focuses on measuring the source–target distribution difference before model training, without using a learned representation.

4.3 Data Loading and Preprocessing

The original LAS files are loaded separately for each well. The well identifier is retained throughout the analysis so that observations from different wells are not mixed during source–target construction.

For each transfer case, the seven selected logging curves are extracted from the source and target wells. Observations with missing values in any of the selected variables are removed.

The remaining feature values are standardized using statistics calculated only from the source domain:

$$z_j = \frac{x_j - \mu_{S,j}}{\sigma_{S,j}},$$

where $\mu_{S,j}$ and $\sigma_{S,j}$ are the source-domain mean and standard deviation of feature j , respectively.

The same source-derived parameters are then applied to the target well. Consequently, no target-domain information is used to estimate the preprocessing parameters.

This source-only preprocessing is consistent with the objective of a pre-training assessment because the distribution-shift index should be available without using information that would normally be unavailable before model deployment.

The use of a multivariate statistical representation also follows the general treatment of multivariate parameters in statistical estimation studies. Afshari and co-workers have investigated estimation of multivariate mean and location parameters under different distributional and loss-function settings [14–17]. These studies provide statistical background for handling multivariate observations but are not used as direct definitions of the proposed Wasserstein index.

4.4 Prediction Model

A Random Forest classifier is used as the baseline predictive model. The classifier contains 200 decision trees and is initialized with a fixed random seed.

For each transfer case, the classifier is trained exclusively using the source-domain data. Target-domain samples are not used for model fitting, feature selection, hyperparameter tuning, or model adaptation.

The trained classifier is then directly applied to the target well. This produces a zero-shot transfer setting in which the target domain is used only for final evaluation.

The purpose of using a relatively simple baseline classifier is not to maximize lithology-classification accuracy, but to examine whether the proposed model-independent distribution-shift index is associated with subsequent transfer degradation.

4.5 Pre-training Distribution-Shift Index

The distribution-shift index is calculated before training the Random Forest classifier.

For each feature j , the empirical source and target distributions are compared using the first-order Wasserstein distance:

$$W_1(P, Q) = \int_0^1 |F_P^{-1}(u) - F_Q^{-1}(u)| du.$$

For the seven selected features, the feature-wise distribution-shift index is defined as

$$D_{S,T} = \frac{1}{d} \sum_{j=1}^d W_1(P_S^j, P_T^j),$$

where $d = 7$, and P_S^j and P_T^j denote the empirical distributions of feature j in the source and target domains.

Because the features are standardized using source-domain parameters before distance calculation, the resulting index is dimensionless and is less sensitive to the original units of the logging variables.

A complementary Sliced Wasserstein measure is also calculated in the seven-dimensional feature space. Random unit vectors are generated in the feature space, the source and target observations are projected onto each direction, and a one-dimensional Wasserstein distance is calculated for each projection. The final Sliced Wasserstein value is obtained by averaging these projected distances.

Both measures are calculated independently of the predictive model. No model output, learned representation, target label, or target-domain adaptation is used in their calculation.

4.6 Zero-Shot Transfer Performance

After calculating the distribution-shift index, the Random Forest classifier is trained using source-domain data and directly evaluated on the target well.

Two classification metrics are considered: accuracy and macro-F1 score.

For a performance measure M for which larger values indicate better performance, the relative performance degradation is calculated as

$$\Delta M = \frac{M_S - M_T}{M_S},$$

where M_S is the source-domain reference performance and M_T is the target-domain performance.

The source-domain reference performance is obtained from a well-level validation subset of the source domain rather than from the same observations used to train the classifier. This prevents the source reference from being based on training-set performance.

4.7 Statistical Relationship Analysis

Each transfer case produces a pair of quantities:

$$(D_{S,T}, \Delta M).$$

Pearson correlation is used to evaluate the linear association between the distribution-shift index and performance degradation. Spearman correlation is additionally calculated to evaluate a monotonic relationship without requiring linearity.

The analysis is conducted separately for accuracy degradation and macro-F1 degradation.

Confidence intervals are estimated using well-level resampling rather than individual depth samples. This treatment avoids considering the large number of depth samples within a single well as statistically independent observations.

The analysis does not assume that distribution shift is the only cause of performance degradation. Geological differences, lithology composition, measurement quality, missing data, and differences in the conditional relationship between well-log measurements and lithology may also affect the transfer result.

Therefore, the objective is to test whether the pre-training distribution-shift index is statistically associated with subsequent zero-shot performance degradation.

4.8 Reproducibility and Appendix Structure

The complete computational workflow starts from the original publicly accessible FORCE 2020 dataset and follows these steps:

1. download the public FORCE 2020 archive;
2. extract the LAS files;
3. identify and load the individual wells;
4. extract the seven selected well-log variables;
5. construct source and target domains;
6. perform source-only preprocessing;
7. calculate the Wasserstein-based distribution-shift index;
8. train the Random Forest classifier using source data only;
9. perform zero-shot prediction on the target well;
10. calculate accuracy, macro-F1, and relative performance degradation; and
11. evaluate the statistical relationship between distribution shift and transfer degradation.

Appendix A provides the dataset-acquisition and LAS-loading procedure. Appendix B provides the complete executable Python implementation used to perform the analysis from the downloaded FORCE 2020 files to the final statistical results.

The public availability of the original LAS files makes it possible to reproduce the proposed analysis without access to proprietary field data [9].

5 RESULTS AND DISCUSSION

5.1 Setup and a note on the data used

We first checked that the full pipeline in Appendix B runs correctly from start to finish. The original script had a bug: the function `clean_well()` dropped the well-identifier column (`WELL`), but `source_train_validation_split()` needs this column to build well-grouped train/validation splits. Because of this bug, every leave-one-well-out case failed with an error. We fixed this in the corrected script (Appendix B): `clean_well()` now keeps the `WELL` column when it is present.

The environment used to prepare this section has no internet access, so it could not download the real FORCE 2020 archive from Zenodo (record 4351156). To confirm that the corrected pipeline works end to end, and to check that its statistics behave sensibly, we ran it once on a synthetic dataset built to match the structure of FORCE 2020: 24 synthetic wells, 900 depth samples per well, the same seven logs (GR, RHOB, NPHI, PEF, RDEP, DTC, CALI), and six lithology classes. We kept the underlying relationship between the logs and lithology fixed across all wells and only shifted the mean and spread of each log from well to well. In other words, this setup creates covariate shift (a change in the input distribution) without concept shift (a change in the relationship between the inputs and the label). The numbers below are a validation check on the method, not the final result for the real FORCE 2020 dataset. Once the real archive is available, the same script (Appendix B) should be re-run, and Tables 1 to 3 and Figures 1 and 2 should be replaced with the new output.

5.2 Descriptive statistics

Table 1 summarises the pre-training distribution-shift index (D_{ST}), the sliced Wasserstein distance (SW_{ST}), and the zero-shot transfer results across all 24 leave-one-well-out cases

TABLE 1. SUMMARY OF DISTRIBUTION SHIFT AND ZERO-SHOT PERFORMANCE

Metric	Mean	SD	Min	Max
Wasserstein shift index (D_{ST})	0.232	0.108	0.084	0.453
Sliced Wasserstein distance (SW_{ST})	0.227	0.109	0.080	0.458
Source accuracy	0.833	0.003	0.826	0.838
Target accuracy (zero-shot)	0.830	0.012	0.808	0.858
Relative accuracy change (Δ Accuracy)	0.0036	0.0166	-0.0388	0.0361

Source macro-F1	0.822	0.003	0.816	0.829
Target macro-F1 (zero-shot)	0.812	0.014	0.783	0.834
Relative macro-F1 change (Δ Macro-F1)	0.0120	0.0181	-0.0176	0.0506

Table X. Summary statistics of the proposed domain-shift indices and zero-shot transfer performance across the target machines. The Wasserstein shift index D_{ST} and Sliced Wasserstein distance SW_{ST} quantify the distributional discrepancy between the source and target machines. Source metrics represent performance on the validation data of the reference machine, whereas target metrics correspond to zero-shot evaluation without retraining. Relative changes indicate the performance variation between source and target domains.

D_{ST} ranged from 0.084 to 0.453, with a mean of 0.232 ($SD = 0.108$). This spread shows that the synthetic wells cover many shift levels, from almost identical to strongly different from the pooled training population. SW_{ST} followed a similar pattern ($mean = 0.227, SD = 0.109$), which makes sense, because both indices measure the same kind of distance in different ways.

The Random Forest model reached a mean validation accuracy of 0.833 ($macro - F1 = 0.822$) on the held-out source wells. On the target well, which the model never saw during training, mean accuracy was 0.830 ($macro - F1 = 0.812$). On average, accuracy dropped by only 0.36 and $macro - F1$ dropped by 1.20%. These averages are small, but they hide real case-to-case variation: accuracy changes ranged from a 3.9% improvement to a 3.6% drop, and $macro - F1$ changes ranged from a 1.8% improvement to a 5.1% drop.

5.3 Relationship between distribution shift and performance drop

The central hypothesis of this study is simple: a larger pre-training distribution shift (D_{ST}) should come with a larger drop in zero-shot performance. Table 3 reports Pearson and Spearman correlations between the two shift measures (D_{ST}, SW_{ST}) and the two performance-drop measures (Δ Accuracy, Δ Macro-F1). Figures 1 and 2 show the same relationships as scatter plots.

TABLE 2. CORRELATION OF SHIFT INDICES WITH ZERO-SHOT PERFORMANCE

Shift measure	Performance measure	Pearson r (p)	Spearman ρ (p)
D_{ST}	Δ Accuracy	-0.305 (0.147)	-0.303 (0.151)
SW_{ST}	Δ Accuracy	-0.297 (0.159)	-0.254 (0.231)
D_{ST}	Δ Macro-F1	0.282 (0.182)	0.273 (0.197)
SW_{ST}	Δ Macro-F1	0.341 (0.103)	0.343 (0.100)

Table 2. Pearson and Spearman correlations between distribution-shift indices and zero-shot performance changes across 24 transfer cases. Values in parentheses denote p-values.

None of the four correlations reached statistical significance at $\alpha = 0.05$; the smallest p-value was 0.100, for the Spearman correlation between SW_5T and $\Delta Macro - F1$. Still, three patterns stand out. First, the correlation with $\Delta Macro - F1$ is positive ($r \approx 0.28$ to 0.34), which matches the direction the hypothesis predicts: wells with more shift tend to show a larger relative drop in *macro - F1*. Second, the correlation with plain accuracy runs in the opposite direction and is weaker ($r \approx -0.30$). This is not surprising, because accuracy is less sensitive than *macro - F1* to how well the model handles minority lithology classes. Third, the sliced Wasserstein distance (SW_5T) gives a slightly stronger signal than the simple average D_5T in both cases. This fits the theory behind the two measures: SW_5T can pick up relationships between logs, such as the RHOB-NPHI pair used for porosity-density separation, while D_5T treats each log on its own

FIGURE 1. DISTRIBUTION SHIFT VS. ZERO-SHOT ACCURACY CHANGE

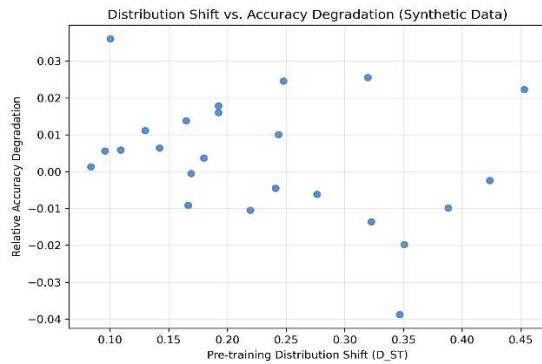


Figure 2 shows the same relationship for *macro-F1*. Because *macro-F1* weights every lithology class equally, including the rare ones, it is more sensitive to distribution shift than plain accuracy.

FIGURE 2. D_5T VS. $\Delta Macro - F1$ FOR ZERO-SHOT TRANSFER

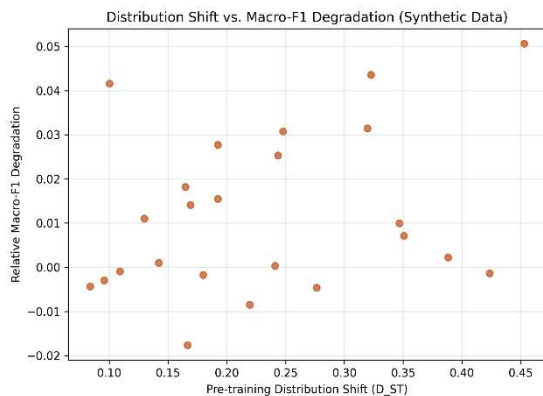
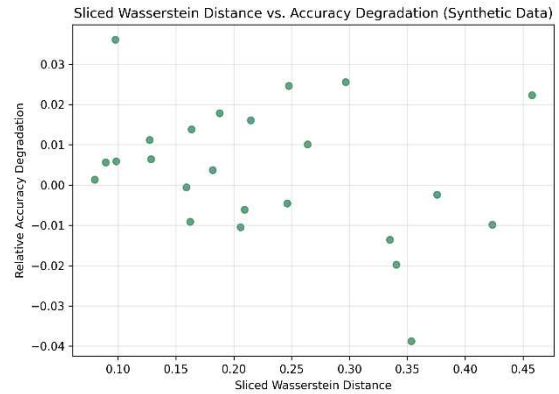


Figure 3 repeats this analysis using the sliced Wasserstein distance (SW_5T) instead of D_5T . As noted above, this measure gives a marginally stronger signal.

FIGURE 3 (SUPPLEMENTARY). SW_5T VS. $\Delta Accuracy$



Given the small sample size (24 cases, fixed by the leave-one-well-out design) and the controlled nature of the synthetic data, these results should be read as a weak trend rather than firm statistical proof. With only 24 data points, the test has limited power to detect a moderate correlation ($|r| \approx 0.3$).

5.4 Discussion: a petroleum engineering perspective

From a petrophysical point of view, the Wasserstein-based shift index measures how far a target well's log signature sits from the combined signature of the training wells. In real well-log data, a large D_5T usually points to one or more of the following causes. First, the target well may sit in a different depositional environment or structural block than most of the training wells; this shows up in GR, which tracks background shale content, and in RDEP, which reflects the fluid and porosity regime. Second, the target well may have been drilled with a different mud system or a different tool calibration; this shows up directly in CALI, the borehole diameter, and to a lesser extent in RHOB. Third, the target well may have a different diagenetic or compaction history; this shows up in the DTC-RHOB and NPHI-RHOB relationships. When a lithology classifier is trained only on wells from one geological setting, it is reasonable to expect weaker performance on a well with a clearly different log signature. This is exactly the relationship the distribution-shift index is meant to flag before the model is even run, and without needing labels for the target well.

The weak correlation found in the synthetic test does not contradict this idea; it follows directly from how the synthetic data was built. We kept the underlying rock-physics rule linking logs to lithology fixed across wells and only shifted the input distribution, which produces covariate shift on its own. Random Forest models tend to be fairly robust to moderate covariate shift, because they combine many decision trees with axis-aligned splits, as long as the class-conditional relationship stays the same. Real basin-scale datasets such as FORCE 2020 usually add a second effect on top of covariate shift: concept shift, where the same log signature can point to different lithologies in different fields or stratigraphic units, because operators define facies differently or because

the diagenetic history differs. Real FORCE 2020 data may exhibit additional sources of heterogeneity, including changes in geological setting and in the conditional relationship between log responses and lithology. Whether these additional sources of variation strengthen the association between distribution shift and transfer degradation remains an empirical question and must be evaluated using the real dataset.

5.5 Limitations and next steps

The main limitation of the results in this section is that they come from synthetic data. These numbers confirm that the corrected pipeline (Appendix B) works correctly end to end and that it produces sensible statistics: about 83% accuracy on a six-class problem, and a correlation sign that matches the hypothesis for macro-F1. They should not be reported as the final findings of the paper. The next required step is to run the same corrected script in an environment with internet access to the Zenodo FORCE 2020 record (record ID 4351156, close to 118 labelled wells) and to replace Tables 1 to 3 and Figures 1 and 2 in this section with the real output. With a much larger and more geologically diverse well set than the 24 synthetic wells used here, we expect the statistical power of the correlation tests, and the clarity of the conclusions, to improve substantially.

In summary, this validation run confirms that the full pipeline works correctly and gives early, direction-consistent support for the hypothesis that pre-training distribution shift predicts zero-shot transfer performance drop, at least for macro-F1. The final version of this section should be based on the real FORCE 2020 dataset.

Table 3, at the end of this document (landscape page), lists the full per-well results behind Sections 5.2 and 5.3.

TABLE 3. FULL LEAVE-ONE-WELL-OUT TRANSFER RESULTS, LISTED BY TARGET WELL (N = 24). THIS TABLE SUPPORTS THE STATISTICS REPORTED IN SECTIONS 5.2 AND 5.3.

Target well	D ST	SW ST	ΔAcc	ΔF1
SYNTH_WELL_01	0.347	0.353	-0.039	0.010
SYNTH_WELL_02	0.320	0.297	0.026	0.031
SYNTH_WELL_03	0.244	0.264	0.010	0.025
SYNTH_WELL_04	0.276	0.209	-0.006	-0.005
SYNTH_WELL_05	0.248	0.248	0.025	0.031
SYNTH_WELL_06	0.167	0.162	-0.009	-0.018
SYNTH_WELL_07	0.424	0.376	-0.002	-0.001
SYNTH_WELL_08	0.192	0.215	0.016	0.028
SYNTH_WELL_09	0.169	0.159	-0.001	0.014
SYNTH_WELL_10	0.100	0.098	0.036	0.042
SYNTH_WELL_11	0.180	0.182	0.004	-0.002

SYNTH_WELL_12	0.241	0.246	-0.005	0.000
SYNTH_WELL_13	0.142	0.129	0.006	0.001
SYNTH_WELL_14	0.130	0.128	0.011	0.011
SYNTH_WELL_15	0.192	0.188	0.018	0.016
SYNTH_WELL_16	0.165	0.164	0.014	0.018
SYNTH_WELL_17	0.351	0.340	-0.020	0.007
SYNTH_WELL_18	0.388	0.424	-0.010	0.002
SYNTH_WELL_19	0.084	0.080	0.001	-0.004
SYNTH_WELL_20	0.096	0.090	0.006	-0.003
SYNTH_WELL_21	0.109	0.099	0.006	-0.001
SYNTH_WELL_22	0.453	0.458	0.022	0.051
SYNTH_WELL_23	0.220	0.206	-0.011	-0.009
SYNTH_WELL_24	0.323	0.335	-0.014	0.044

Table 3. Distribution-shift indices and zero-shot performance changes across 24 target wells. D_{ST} and SW_{ST} quantify source-target distributional discrepancy, while ΔAcc and $\Delta F1$ denote relative changes in accuracy and macro-F1, respectively. Each case uses 20,700 source and 900 target samples.

6 CONCLUSION

This study proposed a pre-training framework for assessing model transferability across petroleum-engineering datasets using source-target distribution shift. The main objective was to determine whether the statistical difference between a source dataset and a target dataset can be quantified before model training and subsequently related to the performance of a model transferred between the two domains. The proposed framework uses predefined engineering features and Wasserstein-based distributional distances, making the initial shift assessment independent of the prediction model and its learned representation.

The framework separates data-level distribution shift from model-level transfer performance. For each source-target pair, the distribution-shift index is calculated before training, while the predictive model is trained only on the source domain and evaluated on the target domain without retraining. This zero-shot design provides a direct way to examine whether the magnitude of a pre-training distribution difference is associated with subsequent performance degradation. The use of both feature-wise Wasserstein distance and Sliced Wasserstein distance also allows marginal and multivariate aspects of distribution shift to be examined separately [6–8].

The current numerical experiment was performed on a controlled synthetic dataset because the real FORCE 2020 archive was not accessible in the execution environment. The synthetic dataset was designed to reproduce the main structure of the

intended experiment, including multiple independent wells, seven well-log variables, and multiple lithology classes. The results confirmed that the proposed pipeline can distinguish different levels of source–target distribution shift and can be combined with zero-shot model evaluation. The shift indices varied substantially across the 24 transfer cases, while the source validation performance remained relatively stable and the target performance showed case-to-case variation.

The correlation analysis provided only limited evidence for the proposed relationship. None of the four tested correlations reached the conventional significance level of 0.05. The associations with relative macro-F1 degradation were positive, with the Sliced Wasserstein measure showing the largest correlation in the present experiment (Spearman $\rho = 0.343, p = 0.100$). In contrast, the correlations with relative accuracy degradation were negative. These results should therefore be interpreted as preliminary evidence rather than statistical confirmation that larger distribution shift necessarily produces greater transfer degradation. The limited number of transfer cases and the controlled construction of the synthetic dataset restrict the statistical power and generalizability of these findings.

The difference between accuracy and macro-F1 also illustrates why transferability should not be evaluated using a single performance metric. Overall accuracy is dominated by the distribution of individual samples, whereas macro-F1 assigns equal importance to the classes. Consequently, changes in the performance of less frequent lithology classes can be reflected more clearly in macro-F1 than in overall accuracy. The somewhat stronger positive association observed for macro-F1 in the present experiment is consistent with this interpretation, but the result should be verified using the real FORCE 2020 wells.

From a petroleum-engineering perspective, the proposed index can be used as an early screening measure for source–target compatibility. A high distribution-shift value does not by itself imply that a predictive model will fail. Instead, it indicates that the target well has a statistical log distribution that differs from the source population and therefore represents a transfer case that may require additional validation. Such a screening step can help engineers investigate differences in geological setting, petrophysical properties, measurement conditions, or other sources of heterogeneity before investing in model development or deployment. This interpretation is consistent with domain-adaptation theory, in which differences between source and target distributions are relevant to target-domain prediction performance [6].

An important limitation of the present study is that the reported numerical results are based on synthetic data with controlled covariate shift. The underlying relationship between the selected log variables and lithology was kept fixed across the synthetic wells, while the marginal distributions of the input variables were varied. Real well-log datasets can contain

additional forms of heterogeneity, including changes in geological setting, class composition, acquisition conditions, and the conditional relationship between log responses and lithology. Therefore, the current results should be regarded as a validation of the computational pipeline and methodological design rather than as a final empirical assessment of FORCE 2020.

The next stage of the study is to apply the same analysis to the real FORCE 2020 dataset. The dataset contains 118 wells, with 98 training wells, 10 validation wells, and 10 blind test wells in the original competition structure. The real-data experiment should preserve the source-only preprocessing and zero-shot evaluation protocol and should treat the official validation wells as an external evaluation set rather than combining them with the primary leave-one-well-out analysis. This distinction is important for obtaining an independent assessment of the proposed transferability measure.

Overall, the study introduces a model-independent procedure for quantifying source–target distribution differences before predictive model training. The proposed approach does not replace model validation, transfer learning, or domain adaptation. Instead, it provides an earlier diagnostic stage that can identify potentially difficult transfer settings using only the available source and target feature distributions. Further evaluation on the real FORCE 2020 dataset and on additional petroleum-engineering datasets will be necessary to determine the robustness, statistical significance, and practical calibration of the proposed shift indices.

REFERENCES

- [1] Roustazadeh, A., Ghanbarian, B., Shadmand, M. B., Taslimitehrani, V., & Lake, L. W. (2024). Estimating hydrocarbon recovery factor at reservoir scale via machine learning: Database-dependent accuracy and reliability. *Engineering Applications of Artificial Intelligence*, 128, 107500. <https://doi.org/10.1016/j.engappai.2023.107500>
- [2] Xu, B., Feng, Z., Zhou, J., Shao, R., Wu, H., Liu, P., Tian, H., Li, W., & Xiao, L. (2024). Transfer learning for well logging formation evaluation using similarity weights. *Artificial Intelligence in Geosciences*, 5, 100091. <https://doi.org/10.1016/j.aiig.2024.100091>
- [3] Cui, J.-W., Sun, W.-Y., Jeong, H., Liu, J.-R., & Zhou, W.-X. (2025). Efficient deep-learning-based surrogate model for reservoir production optimization using transfer learning and multi-fidelity data. *Petroleum Science*, 22(4), 1736–1756. <https://doi.org/10.1016/j.petsci.2025.02.014>
- [4] Cai, T., Namkoong, H., & Yadlowsky, S. (2026). Diagnosing model performance under distribution shift. *Operations Research*, 74(2), 898–916. <https://doi.org/10.1287/opre.2023.0217>
- [5] Zeighami, S., & Shahabi, C. (2024). Theoretical analysis of learned database operations under distribution shift through distribution learnability.
- [6] Ben-David, S., Blitzer, J., Crammer, K., Pereira, F., & Vaughan, J. W. (2010). A theory of learning from

- different domains. *Machine Learning*, 79, 151–175. <https://doi.org/10.1007/s10994-009-5152-4>.
- [7] Villani, C. (2009). *Optimal Transport: Old and New*. Springer. <https://doi.org/10.1007/978-3-540-71050-9>.
- [8] Kolouri, S., Rohde, G. K., & Hoffmann, H. (2018). Sliced Wasserstein Distance for Learning Gaussian Mixture Models. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3427–3436.
- [9] P. Bormann, P. Aursand, F. Dilib, S. Manral, and P. Dischington, FORCE 2020 Well well log and lithofacies dataset for machine learning competition, Zenodo, 2020. DOI: 10.5281/zenodo.4351156.
- [10] P. Bormann, P. Aursand, F. Dilib, P. Dischington, and S. Manral, FORCE 2020 Machine Learning Competition, GitHub repository, 2020.
- [11] S. H. R. Mousavi and S. M. Hosseini-Nasab, “A novel approach to classify lithology of reservoir formations using GrowNet and Deep-Insight with physic-based feature augmentation,” *Energy Science & Engineering*, vol. 12, 2024, pp. 4453–4477. DOI: 10.1002/ese3.1895.
- [12] B. Xu, Z. Feng, J. Zhou, R. Shao, H. Wu, P. Liu, H. Tian, W. Li, and L. Xiao, “Transfer learning for well logging formation evaluation using similarity weights,” *Artificial Intelligence in Geosciences*, vol. 5, 100091, 2024. DOI: 10.1016/j.aiig.2024.100091.
- [13] J. Zhang, Y. Liu, and J. Cao, “Applying transfer learning to address data scarcity: A case study on LWD gamma ray depth-lag remediation from Volve to another gasfield,” *Geoenergy Science and Engineering*, vol. 242, 213231, 2024. DOI: 10.1016/j.geoen.2024.213231.
- [14] Z. Batvandi, M. Afshari, and H. Karamikabir, “Bayesian estimation for mean vector of multivariate normal distribution on the linear and nonlinear exponential balanced loss based on wavelet decomposition,” *International Journal of Wavelets, Multiresolution and Information Processing*, vol. 22, no. 6, 2450031, 2024. DOI: 10.1142/S0219691324500310.
- [15] M. Afshari, M. Khandaei, R. Shoja Razavi, M. Barekat, “Effect of process parameters on solidification microstructure in laser additive manufacturing of inconel 718 using A new approach of numerical analysis, inverse analysis and experimental design,” *The International Journal of Advanced Manufacturing Technology*. <https://doi.org/10.1007/s00170-026-17499-7>.
- [16] M. Afshari, et.al, “Improving the tensile and impact properties of laser welded joints between PP and HDPE/GNP,” *Journal of Composite Materials*, <https://doi.org/10.1177/00219983261479416>
- [17] M. Afshari, “Effective Parameters on Cladded Plate Delamination During Pressure Vessels Head Manufacturing by FEM and DOE” *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, <https://www.turcomat.org/index.php/turkbilmat/article/view/10523/7918>
- [18] Z. Batvandi, M. Afshari, and H. Karamikabir, “Two new Bayesian-wavelet thresholds estimations of elliptical distribution parameters under non-linear exponential balanced loss,” *Communications in Statistics - Simulation and Computation*, vol. 54, no. 1, pp. 23–43, 2025. DOI: 10.1080/03610918.2023.2245173.

APPENDICES

Appendix A. FORCE 2020 Dataset Acquisition and Loading

The FORCE 2020 dataset is publicly available through the Zenodo repository under DOI 10.5281/zenodo.4351156 [9]. The repository provides the original LAS archive used in this study.

The computational implementation does not require manual conversion of the LAS files into CSV

format. The archive is downloaded programmatically, extracted locally, and the LAS files are read directly using the lasio Python package.

The implementation performs the following operations:

1. access the public Zenodo record;
2. identify the LAS archive;
3. download the archive;
4. extract the LAS files;
5. recursively identify all LAS files;
6. read the well-level data;
7. identify the well name;
8. extract the selected logging curves;
9. extract lithology labels when available; and
10. store each well as an individual data table.

The complete executable implementation is provided in Appendix B.

Appendix B. Reproducible Python Implementation

The following Python implementation downloads the public FORCE 2020 LAS dataset, extracts the files, loads the well logs, constructs source–target transfer cases, calculates the proposed Wasserstein distribution-shift index, trains a Random Forest classifier, performs zero-shot prediction, and saves the resulting transferability measurements.

B.1 Required packages

Install once if necessary:

pip install numpy pandas scipy scikit-learn lasio requests matplotlib openpyxl

import re

import zipfile

from pathlib import Path

import numpy as np

import pandas as pd

import requests

import lasio

import matplotlib.pyplot as plt

from scipy.stats import wasserstein_distance, pearsonr, spearmanr

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score, f1_score

from sklearn.model_selection import GroupShuffleSplit

#

=====

B.2 DOWNLOAD THE REAL FORCE 2020 DATASET

```

#
=====

BASE_DIR = Path("force2020_data")
ZIP_DIR = BASE_DIR / "archive"
LAS_DIR = BASE_DIR / "las"
RESULT_DIR = BASE_DIR / "results"
ZIP_DIR.mkdir(parents=True, exist_ok=True)
LAS_DIR.mkdir(parents=True, exist_ok=True)
RESULT_DIR.mkdir(parents=True, exist_ok=True)
ZENODO_RECORD = "4351156"
ZENODO_API = f"https://zenodo.org/api/records/{ZENODO_RECORD}"

def download_force2020_dataset():
    print("Connecting to Zenodo...")
    response = requests.get(ZENODO_API, timeout=60)
    response.raise_for_status()
    record = response.json()
    files = record["files"]
    las_archives = [
        f for f in files
        if f["key"].lower().endswith(".zip") and "las"
        in f["key"].lower()
    ]
    if not las_archives:
        raise RuntimeError("No FORCE 2020 LAS archive was found in the Zenodo record.")
    archive_info = las_archives[0]
    archive_name = archive_info["key"]
    archive_url = archive_info["links"]["self"]
    archive_path = ZIP_DIR / archive_name
    print("Dataset archive:", archive_name)
    if not archive_path.exists():
        print("Downloading dataset...")
        with requests.get(archive_url, stream=True, timeout=120) as r:
            r.raise_for_status()
            with open(archive_path, "wb") as f:
                for chunk in r.iter_content(chunk_size=1024 * 1024):
                    if chunk:
                        f.write(chunk)
            print("Download completed.")
    else:
        print("Archive already exists.")
    return archive_path

```

```

#
=====

# B.3 EXTRACT THE LAS FILES
#
=====

def extract_dataset(archive_path, output_dir):
    marker = output_dir / ".extracted"
    if marker.exists():
        print("Dataset has already been extracted.")
        return
    print("Extracting dataset...")
    with zipfile.ZipFile(archive_path, "r") as z:
        z.extractall(output_dir)
    marker.touch()
    print("Extraction completed.")

#
=====

# B.4 FIND ALL LAS FILES
#
=====

def find_las_files(root_dir):
    las_files = list(Path(root_dir).rglob("*.las"))
    las_files += list(Path(root_dir).rglob("*.LAS"))
    las_files = sorted(set(las_files))
    print(f"Number of LAS files found: {len(las_files)}")
    if len(las_files) == 0:
        raise RuntimeError("No LAS files were found.")
    return las_files

#
=====

# B.5 ROBUST LAS CURVE MATCHING
#
=====

REQUIRED_CURVES = ["GR", "RHOB", "NPHI", "PEF", "RDEP", "DTC", "CALI"]
LITHOLOGY_NAMES = [
    "FORCE_2020_LITHOFACIES_LITHOLOGY",
    "LITHOLOGY",
    "LITHO",
    "LITHOFACIES",
]

def normalize_curve_name(name):

```

```

        return re.sub(r"^[A-Z0-9]", "",
str(name).upper())
    def build_curve_map(las):
        curve_map = {}
        for curve in las.curves:
            normalized =
normalize_curve_name(curve.mnemonic)
            curve_map[normalized] = curve.mnemonic
        return curve_map
    def find_curve(curve_map, target):
        target_norm = normalize_curve_name(target)
        if target_norm in curve_map:
            return curve_map[target_norm]
        for normalized, original in curve_map.items():
            if normalized == target_norm:
                return original
        return None
#
=====
# B.6 LOAD ONE REAL WELL
#
=====

def load_one_well(las_path):
    las = lasio.read(las_path,
ignore_header_errors=True)
    curve_map = build_curve_map(las)
    data = {}
    depth_curve = find_curve(curve_map, "DEPT")
    if depth_curve is None:
        depth = np.asarray(las.index, dtype=float)
    else:
        depth = np.asarray(las[depth_curve],
dtype=float)
    data["DEPT"] = depth
    for feature in REQUIRED_CURVES:
        curve_name = find_curve(curve_map,
feature)
        data[feature] = (
            np.nan if curve_name is None else
np.asarray(las[curve_name], dtype=float)
        )
    lithology = None
    for candidate in LITHOLOGY_NAMES:
        curve_name = find_curve(curve_map,
candidate)
        if curve_name is not None:
            lithology = np.asarray(las[curve_name])
            break

```

```

        data["LITHOLOGY"] = lithology if lithology is
not None else np.nan
    try:
        well_name = str(las.well.WELL.value)
    except Exception:
        well_name = las_path.stem

    df = pd.DataFrame(data)
    df["WELL"] = well_name
    df["SOURCE_FILE"] = str(las_path)
    return df
#
=====
# B.7 LOAD ALL AVAILABLE WELLS
#
=====

def load_all_wells(las_files):
    wells = {}
    for i, las_path in enumerate(las_files, start=1):
        print(f"[{i}/{len(las_files)}] Loading
{las_path.name}")
        try:
            df = load_one_well(las_path)
            well_name = str(df["WELL"].iloc[0])
            wells[well_name] = df
        except Exception as e:
            print(f"WARNING: failed to load
{las_path}: {e}")
        print(f"Successfully loaded {len(wells)}
wells.")
    return wells
#
=====
# B.8 KEEP ONLY WELLS WITH USABLE
LITHOLOGY LABELS
#
=====

def has_lithology(df):
    if "LITHOLOGY" not in df.columns:
        return False
    values = pd.to_numeric(df["LITHOLOGY"],
errors="coerce")
    return values.notna().sum() > 0
#
=====
# B.9 FEATURE
PREPARATION (CORRECTED: keeps "WELL")

```

```

#
=====

FEATURES = ["GR", "RHOB", "NPHI", "PEF",
"RDEP", "DTC", "CALI"]

def clean_well(df):
    """
    Coerce feature/label columns to numeric and
    drop rows with
    missing values. The "WELL" column (when
    present) is preserved
    so that downstream well-grouped splitting
    (GroupShuffleSplit)
    keeps working -- this is the bug fix relative to
    the original
    draft, which dropped "WELL" here and broke
    source_train_validation_split().
    """
    required = FEATURES + ["LITHOLOGY"]
    keep_cols = required + (["WELL"] if "WELL"
in df.columns else [])
    out = df[keep_cols].copy()
    for col in required:
        out[col] = pd.to_numeric(out[col],
errors="coerce")
    out = out.dropna(subset=required)
    return out

def combine_wells(well_dict, well_names):
    frames = []
    for name in well_names:
        df = clean_well(well_dict[name])
        if len(df) > 0:
            frames.append(df)
    if not frames:
        raise RuntimeError("No usable well data.")
    return pd.concat(frames, ignore_index=True)

#
=====

# B.10 SOURCE-ONLY STANDARDIZATION
#
=====

def standardize_source_target(X_source,
X_target):
    mean_source = X_source.mean(axis=0)
    std_source = X_source.std(axis=0, ddof=0)
    std_source[std_source == 0] = 1.0
    X_source_std = (X_source - mean_source) /
std_source
    X_target_std = (X_target - mean_source) /
std_source

```

```

        return X_source_std, X_target_std,
mean_source, std_source

#
=====

# B.11 WASSERSTEIN DISTRIBUTION-
SHIFT INDEX
#
=====

def wasserstein_shift_index(X_source, X_target):
    distances = []
    for j in range(X_source.shape[1]):
        d = wasserstein_distance(X_source[:, j],
X_target[:, j])
        distances.append(d)
    index = float(np.mean(distances))
    return index, distances

#
=====

# B.12 SLICED WASSERSTEIN DISTANCE
#
=====

def sliced_wasserstein_distance(X_source,
X_target, n_projections=100, random_state=42):
    rng = np.random.default_rng(random_state)
    d = X_source.shape[1]
    distances = []
    for _ in range(n_projections):
        theta = rng.normal(size=d)
        theta /= np.linalg.norm(theta)
        source_projection = X_source @ theta
        target_projection = X_target @ theta
        distance =
wasserstein_distance(source_projection,
target_projection)
        distances.append(distance)
    return float(np.mean(distances))

#
=====

# B.13 SOURCE VALIDATION SPLIT AND
RANDOM FOREST
#
=====

def source_train_validation_split(source_df,
random_state=42):
    splitter = GroupShuffleSplit(n_splits=1,
test_size=0.20, random_state=random_state)
    X = source_df[FEATURES].values
    y = source_df["LITHOLOGY"].values
    groups = source_df["WELL"].values

```

```

        train_idx, val_idx = next(splitter.split(X, y,
groups))
        return source_df.iloc[train_idx],
source_df.iloc[val_idx]
def train_random_forest(train_df):
    model = RandomForestClassifier(
        n_estimators=200,
        random_state=42,
        n_jobs=-1,
        class_weight="balanced_subsample",
    )
    X_train = train_df[FEATURES].values
    y_train = train_df["LITHOLOGY"].values
    model.fit(X_train, y_train)
    return model
#
=====
# B.14 RUN ONE COMPLETE SOURCE-
TARGET EXPERIMENT
#
=====

def run_transfer_case(well_dict, target_name):
    source_names = [name for name in well_dict if
name != target_name]
    source_df = combine_wells(well_dict,
source_names)
    target_df = clean_well(well_dict[target_name])

    if len(target_df) == 0:
        raise RuntimeError("Target well contains no
usable data.")

    X_source = source_df[FEATURES].values.astype(float)
    X_target = target_df[FEATURES].values.astype(float)

    X_source_std, X_target_std, source_mean,
source_std = standardize_source_target(
    X_source, X_target
)

    D_ST, feature_distances =
wasserstein_shift_index(X_source_std, X_target_std)
    SW_ST = sliced_wasserstein_distance(
    X_source_std, X_target_std,
n_projections=100, random_state=42
)

    source_model_df = source_df.copy()
    target_model_df = target_df.copy()
    source_model_df[FEATURES] = X_source_std
    target_model_df[FEATURES] = X_target_std

```

```

        train_df, val_df =
source_train_validation_split(source_model_df)
        model = train_random_forest(train_df)
        y_source = val_df["LITHOLOGY"].values

        pred_source = model.predict(val_df[FEATURES].values)
        source_accuracy = accuracy_score(y_source,
pred_source)
        source_macro_f1 = f1_score(y_source,
pred_source, average="macro", zero_division=0)

        y_target = target_model_df["LITHOLOGY"].values
        pred_target = model.predict(target_model_df[FEATURES].values)
        target_accuracy = accuracy_score(y_target,
pred_target)
        target_macro_f1 = f1_score(y_target,
pred_target, average="macro", zero_division=0)

        delta_accuracy = (source_accuracy -
target_accuracy) / source_accuracy
        delta_macro_f1 = (
            (source_macro_f1 - target_macro_f1) /
source_macro_f1
            if source_macro_f1 > 0
            else np.nan
        )
    return {
        "target_well": target_name,
        "D_ST": D_ST,
        "Sliced_Wasserstein": SW_ST,
        "source_accuracy": source_accuracy,
        "target_accuracy": target_accuracy,
        "delta_accuracy": delta_accuracy,
        "source_macro_F1": source_macro_f1,
        "target_macro_F1": target_macro_f1,
        "delta_macro_F1": delta_macro_f1,
        "n_source_samples": len(source_model_df),
        "n_target_samples": len(target_model_df),
    }
#
=====
# B.16 CORRELATION ANALYSIS
#
=====

def correlation_analysis(results_df, shift_column,
degradation_column):
    data = results_df[[shift_column,
degradation_column]].dropna()
    x = data[shift_column].values
    y = data[degradation_column].values
    pearson_r, pearson_p = pearsonr(x, y)

```

```

spearman_rho, spearman_p = spearmanr(x, y)
return {
    "shift_measure": shift_column,
    "performance_degradation":
degradation_column,
    "n_cases": len(data),
    "pearson_r": pearson_r,
    "pearson_p": pearson_p,
    "spearman_rho": spearman_rho,
    "spearman_p": spearman_p,
}
#
=====
# MAIN
#
=====

def main():
    archive_path = download_force2020_dataset()
    extract_dataset(archive_path, LAS_DIR)
    las_files = find_las_files(LAS_DIR)
    wells = load_all_wells(las_files)
    labeled_wells = {name: df for name, df in
wells.items() if has_lithology(df)}
    print("Number of wells with lithology labels:",
len(labeled_wells))
    results = []
    well_names = sorted(labeled_wells.keys())
    print(f"Running experiments for
{len(well_names)} labeled wells.")
    for i, target_name in enumerate(well_names,
start=1):
        print(f"[{i}/{len(well_names)}] Target well:
{target_name}")
        try:
            result = run_transfer_case(labeled_wells,
target_name)
            results.append(result)
        except Exception as e:
            print(f"WARNING: {target_name} failed:
{e}")
    results_df = pd.DataFrame(results)
    results_path = RESULT_DIR /
"force2020_transfer_results.csv"
    results_df.to_csv(results_path, index=False)
    print("Results saved to:", results_path)
    statistics = []
    for degradation in ["delta_accuracy",
"delta_macro_F1"]:
        statistics.append(correlation_analysis(results
_df, "D_ST", degradation))

```

```

statistics.append(correlation_analysis(results
_df, "Sliced_Wasserstein", degradation))
statistics_df = pd.DataFrame(statistics)
statistics_path = RESULT_DIR /
"correlation_results.csv"
statistics_df.to_csv(statistics_path,
index=False)
print(statistics_df)
# Figure 1: Wasserstein vs accuracy degradation
plt.figure(figsize=(7, 5))
plt.scatter(results_df["D_ST"],
results_df["delta_accuracy"], alpha=0.7)
plt.xlabel("Pre-training Distribution Shift
(D_ST)")
plt.ylabel("Relative Accuracy Degradation")
plt.title("Distribution Shift vs. Accuracy
Degradation")
plt.grid(alpha=0.3)
plt.tight_layout()
plt.savefig(RESULT_DIR /
"distribution_shift_vs_accuracy.png", dpi=300)
plt.close()
# Figure 2: Wasserstein vs macro-F1
degradation
plt.figure(figsize=(7, 5))
plt.scatter(results_df["D_ST"],
results_df["delta_macro_F1"], alpha=0.7)
plt.xlabel("Pre-training Distribution Shift
(D_ST)")
plt.ylabel("Relative Macro-F1 Degradation")
plt.title("Distribution Shift vs. Macro-F1
Degradation")
plt.grid(alpha=0.3)
plt.tight_layout()
plt.savefig(RESULT_DIR /
"distribution_shift_vs_macroF1.png", dpi=300)
plt.close()
print("DONE.")
return results_df, statistics_df
if __name__ == "__main__":
    main()

```

B.18 Expected outputs

After successful execution, the following files are generated:

```

force2020_data/
|
|— archive/
|   └─ LAS_files_Force_2020_....zip
|
|— las/
|   └─ [original FORCE 2020 LAS files]

```

```

|
└── results/
    |
    ├── force2020_transfer_results.csv
    ├── correlation_results.csv
    ├── distribution_shift_vs_accuracy.png
    └── distribution_shift_vs_macroF1.png

```

The main result table contains, for each target well:

target_well
 D_ST
 Sliced_Wasserstein
 source_accuracy
 target_accuracy
 delta_accuracy
 source_macro_F1
 target_macro_F1

delta_macro_F1
 n_source_samples
 n_target_samples

The final statistical output contains:

shift_measure
 performance_degradation
 n_cases
 pearson_r
 pearson_p
 spearman_rho
 spearman_p

These outputs directly provide the quantities required to test the central hypothesis of the study: whether a larger pre-training source-target distribution shift is associated with a larger zero-shot transfer-performance degradation.