Data Science

Shahid Beheshti University
Spring 2025
Regression Challenge

Assignment #5

1 Theoretical

Answer the following questions. Justify all answers with clear reasoning or computation.

Q1. (Feature Engineering Effectiveness)
Why is feature engineering considered a creative process in data science, and how should teams approach it effectively?

Q2. (Categorical Encoding Methods)

(a) Explain One-hot, Label, Target, and Hash encoding. Compare them.

(b) What are the limitations of One-hot encoding, especially when dealing with high-cardinality categorical features?

(c) In what scenarios is label encoding more appropriate than One-hot encoding, and why?

(d) How can overfitting be avoided when applying target encoding?

(e) How does hash encoding address the sparsity problem in categorical variables, and what potential issue does it introduce?

Q3. (Category Embeddings)
How do category embeddings work, and why might they outperform traditional encoding methods for high-cardinality features?

Q4. (Rounding Numerical Features)
What is the benefit of rounding numerical features before using them in a model, and what is a risk of doing so?

Q5. (Non-linearities and Cyclical Features)

(a) How can interaction features or polynomial encodings help a linear model solve non-linear problems?

(b) When engineering features from date/time data, what techniques ensure that models handle cyclical variables correctly?

Q6. (TF-IDF and Dimensionality Reduction)
What role does TF-IDF play in natural language processing, and how can dimensionality reduction improve model performance with text data?

Q7. (Neural Networks and Feature Engineering)
Why do Neural Networks and Deep Learning models often require less manual feature engineering compared to traditional machine learning models?

2 Practical

2.1 Introduction

In this assignment, you will act as data scientists for a fictitious insurance company. Your mission is to build a regression model that accurately predicts the **Policy Cost** based on a diverse set of anonymized customer and policy-related features. The dataset mimics real-world insurance data, making the task a comprehensive exercise in handling various data types and predicting continuous values.

2.2 Dataset and Competition Setup

You have three files on This Kaggle competition:

train.csv contains approximately 1,100,000 rows, each with an integer `ID`, numerous anonymized features (e.g., `Years Lived`, `Sex`, `Yearly Earnings`, etc.), and the target `Policy Cost`. **x_test.csv** mirrors that structure but omits the `Policy Cost`, holding 100,000 rows that will decide the leaderboard. **sample_submission.csv** is a two-column template (`ID`, `Policy Cost`).

After logging in, you may submit up to thirty prediction files per calendar day. Kaggle shows a public score based on 50% of the hidden labels (approximately 50,000 rows); the private score—used for marking—remains concealed until the deadline, also based on 50% of the hidden labels (approximately 50,000 rows).

Your bonus score will be computed by the following formula and added directly to your practical grade!

$$\text{Bonus} \;=\; \frac{1}{1 + 0.5 \times (\text{rank} - 1)}$$

2.3 What You Must Deliver

Your work has two facets. First, a *fully reproducible* notebook (Jupyter or Colab) that loads the data, explores it, trains at least one solid model, and writes a submission file. Second, a concise PDF report that we can read without running any code. The report should narrate your decisions, highlight discoveries, and justify the final architecture. When grading we will retrain your best model offline, so keep random seeds fixed and explain any non-deterministic step.

2.4 Suggested Workflow

Begin with a light exploratory analysis: inspect the distribution of the `Policy Cost`, plot feature distributions, handle missing values (imputation), and calculate simple correlation or mutual-information statistics to understand relationships between features and the target.

Next, construct baseline learners. Linear regression is a welcome starting point, but do not stop there. Try at least two other families suitable for regression: Support Vector Regressors (SVR, linear or RBF kernel), Decision Trees pruned for generalisation (sometimes called a "good tree"), or ensemble methods such as Random Forests, Gradient Boosting Machines (e.g., XGBoost, LightGBM, CatBoost). k-nearest-neighbour regressors and Gaussian Process Regressors are also permitted. **Neural networks are strictly off-limits for the primary prediction task** in this exercise. The only exception is if a

neural network is utilized solely for **data augmentation** purposes. Our aim is to practise classic, interpretable tabular ML and powerful tree-based ensemble methods.

2.5 Feature Engineering Guidance

Feature engineering is a critical step to boost your model's performance. Consider the following techniques:

- **Temporal Features from 'Coverage Commencement'**: Extract components from the 'Coverage Commencement' column. For example:
  - Year, Month, Day of Week, Day of Year.
  - Season (e.g., Spring, Summer, Autumn, Winter).
  - Elapsed time since policy start (e.g., Policy Age at prediction time, if a reference date is provided).
  - Time differences: If multiple date columns were available, calculating the duration between events could be valuable.
- **Numerical Feature Transformations**:
  - **Binning/Discretization**: Group continuous numerical features (e.g., 'Years Lived', 'Yearly Earnings', 'Wellness Index') into bins. This can help capture non-linear relationships.
  - **Logarithmic/Power Transformations**: Apply transformations (e.g., $\log(x + 1)$, $\sqrt{x}$) to skewed numerical features like 'Yearly Earnings' or 'Premium Amount' to make their distribution more normal, which can benefit some models.
  - **Interaction Terms**: Create new features by multiplying or dividing existing numerical features (e.g., 'Years Lived' × 'Prior Claims' or 'Yearly Earnings' / 'Dependent Count').
- **Categorical Feature Engineering**:
  - **Combining Rare Categories**: For categorical features with many unique values but some rare ones, consider grouping infrequent categories into an 'Other' category to reduce dimensionality and noise.
  - **Categorical Interaction Terms**: Create new categorical features by combining two or more existing categorical features (e.g., 'Sex' + 'Relationship Status'). This can capture specific interactions.
  - **Target/Mean Encoding (Use with Caution!)**: Replace categorical labels with the mean of the target variable for that category. This can be powerful but requires careful cross-validation to prevent data leakage.
- **Feature Scaling**:
  - While tree-based models are often less sensitive to feature scaling, other models like Support Vector Regressors (SVR) or linear models require features to be scaled (e.g., using 'StandardScaler' or 'MinMaxScaler') to perform optimally. Consider applying scaling if you venture beyond tree-based ensembles.

2.6 Model Interpretation

Numbers are not enough. Therefore, reserve a slice of your analysis for feature importance. Permutation tests, impurity measures in tree ensembles, or SHAP values all earn credit. Summarise your findings in plain language: *"Features related to financial rating and vehicle age are the strongest drivers for policy cost …"*. Finish with two concrete suggestions for how such knowledge could guide data collection or business strategies.

2.7 Evaluation and Minimum Performance

Kaggle's metric is **Root Mean Squared Logarithmic Error (RMSLE)** on the leaderboard. To understand how RMSLE is calculated, refer to the 'Evaluation' tab on the competition page. To pass the assignment, your best private RMSLE score must be at most **1.55**. Any solution above that line receives partial credit only. To participate in the bonus score calculation, your best private RMSLE score must be at most **1.45**.

3 Submission

Upload (1) the notebook; (2) the PDF report; and your answers to theoretical part to the course classroom (Google Classroom).