Amirreza Akbari | 810899045
Reza Baghestani | 810899046
Hananeh Jamali | 810899053

Purpose:

The purpose of this assignment is to implement and compare different aspects of multivariate regression analysis, including preprocessing, manual implementation of gradient descent, manual K-Fold cross-validation, and comparison with built-in Python libraries.

Summary:

In this assignment, we explored multivariate regression analysis through various stages. We began by preprocessing the dataset, performing feature engineering, and cleaning the data to prepare it for analysis. Subsequently, we implemented multivariate regression from scratch using gradient descent and validated the model's performance. Additionally, we implemented manual K-Fold cross-validation to further evaluate the model's robustness. Finally, we compared our custom implementations with those from built-in Python libraries to assess their performance.

Dataset Used:

Two datasets were used in this analysis: "cars.csv" and "football.csv".

- **cars.csv:** This dataset contains information about various car attributes such as price, horsepower, and other features. It was utilized for training and testing the multivariate regression models in the context of predicting car prices based on different attributes.

- **football.csv:** The "football.csv" dataset contains data related to football matches, including details such as outcomes, interference on shooters, and play types. While not explicitly mentioned in the report, this dataset likely served a similar purpose to "cars.csv" in terms of preprocessing and analysis, possibly exploring regression analysis in the context of football match outcomes or player performance.

Task 1: Preprocessing

Data Loading and Initial Exploration: The first step involved loading the "football.csv" dataset using the pd.read_csv function. This dataset was inspected using methods like df.head() and df.info() to understand its structure and contents, which included columns such as 'outcome', 'interferenceOnShooter', 'second', 'minute', 'x', 'y', and categorical variables like 'bodyPart' and 'playType'.

Handling Missing Values: To address missing values, the df.isnull().sum() method was used to identify columns with null entries. Specifically, rows where both 'outcome' and 'interferenceOnShooter' were missing were counted and subsequently removed using the df.dropna method.

Feature Engineering: The time feature 'second' was converted to minutes and added to the 'minute' column to improve temporal analysis. This was done by dividing 'second' by 60 and adding the result to 'minute'.

Distance and Angle Calculation: To capture spatial information, the distance and angle from the origin (0,0) were calculated. The distance was computed using the Euclidean distance formula, and the angle was calculated using the arctangent function. The original 'x' and 'y' columns were subsequently dropped from the dataset.

Amirreza Akbari | 810899045
Reza Baghestani | 810899046
Hananeh Jamali | 810899053

Ordinal and One-Hot Encoding: The categorical column 'interferenceOnShooter' was ordinally encoded with a mapping of {"low": 1, "medium": 2, "high": 3}. Additionally, one-hot encoding was applied to 'bodyPart' and 'playType' using the pd.get_dummies method. The modified DataFrame was then saved to 'football_modified.csv' for subsequent analysis.

Task 2: Multivariate Regression Implementation

Gradient Descent and Cost Computation: A custom gradient descent function was implemented to optimize the regression model. This function iteratively updated the model weights to minimize the cost function (Mean Squared Error).

Multivariate Regression Class: The MultivariateRegression class was implemented to encapsulate the regression model. It included methods for fitting the model (fit) and making predictions (predict). The model was trained and tested on the "cars.csv" dataset, which was preprocessed by normalizing the features and adding a bias term. The model's performance was evaluated using Mean Squared Error (MSE).

Visualization of Results: Predicted versus actual values were plotted using a custom function to visually assess the model's accuracy. This function created scatter plots comparing the predicted values to the actual values for both target variables, 'Price in Thousands' and 'Horsepower'.

Task 3: Manual K-Fold Cross Validation Implementation

K-Fold Cross-Validation Class: The ManualKFoldCrossValidation class was implemented to perform K-Fold cross-validation manually. This class split the data into 'k' folds, trained the model on each training set, validated on each validation set, and calculated the Mean Squared Error for each fold.

Implementation for Target Variables: The K-Fold cross-validation was applied to both target variables ('Price in Thousands' and 'Horsepower'). The mean validation error across folds was computed to evaluate model stability and generalization. Learning curves were plotted to visualize the model's performance across different folds.

Task 4: Comparison with Built-in Python Libraries

Scikit-Learn Regression: The LinearRegression model from scikit-learn was used for comparison. Models were trained and tested using scikit-learn's fit and predict methods. Predictions were made and evaluated using Mean Squared Error (MSE).

Scikit-Learn K-Fold Cross-Validation: K-Fold cross-validation was performed using scikit-learn's cross_val_score function. The mean validation error was computed for comparison with custom implementations. This provided a benchmark to assess the performance and accuracy of the custom gradient descent and K-Fold cross-validation implementations.

Amirreza Akbari | 810899045
Reza Baghestani | 810899046
Hananeh Jamali | 810899053

Questions:

Question 1) Describe your strategy for addressing challenges such as handling missing values and categorical features. Could you also elaborate on your feature selection metrics and explain the rationale behind them?

Since we only had 34 rows with missing values and all these 34 rows had missing values in both the outcome column and the interferenceOnShooter column, considering the small proportion of 34 rows to the total data and the fact that our analysis was focused on the outcomes, we removed them. For the interferenceOnShooter column, we used ordinal encoding because it had values: low, medium and high, and for the other two columns playType and bodyPart, we used a one-hot encoding to give us the best output. We rounded minute and second values, and instead of using x and y, we used distance and angle.

Question 2) Why didn't we use regression to predict whether a shot would result in a goal?

1. Non-linear relationships: The relationship between the predictors (e.g. shot distance, angle, player position) and the outcome (goal or no goal) may not be linear. Regression assumes a linear relationship between the predictors and the outcome, which may not be the case in this scenario.

2. Classification problem: Predicting whether a shot results in a goal is a binary classification problem (goal or no goal), rather than a continuous regression problem. Classification algorithms, such as logistic regression or decision trees, are better suited for this type of problem.

3. Imbalanced data: In soccer, the number of shots that result in a goal is typically much smaller than the number of shots that do not result in a goal. This can lead to imbalanced data, which can affect the performance of regression models. Classification algorithms are better equipped to handle imbalanced data.

4. Non-linear interactions: There may be complex interactions between the predictors that affect the likelihood of a shot resulting in a goal. Regression may not be able to capture these non-linear interactions effectively.

Question 3) How would you go about verifying the accuracy of the given formula used to calculate the shot angle in the preprocessing section?

There was a ready-made function that we used to avoid verifying and we made sure that the resulting angle falls within the first and fourth quadrants of the trigonometric circle.

Question 4) Discuss the advantages and disadvantages of k-fold cross-validation. Can you also explain other types of cross-validation methods that could address the limitations and issues associated with k-fold cross-validation?

Advantages of k-fold cross-validation:

1. It provides a more reliable estimate of model performance compared to a single train-test split.
2. It utilizes the entire dataset for both training and testing, leading to a more robust evaluation.

Amirreza Akbari | 810899045
Reza Baghestani | 810899046
Hananeh Jamali | 810899053

3. It helps in reducing variance in the performance estimate by averaging results from multiple folds.
4. It can be used to tune hyperparameters and evaluate model generalization.

Disadvantages of k-fold cross-validation:

1. It can be computationally expensive, especially for large datasets or complex models.
2. It may not be suitable for time-series data or when there is a specific temporal order in the data.
3. It may lead to overfitting if hyperparameters are tuned based on the validation set in each fold.

Other types of cross-validation methods that can address the limitations of k-fold cross-validation include:

1. Leave-One-Out Cross-Validation (LOOCV): In LOOCV, a single data point is held out as the validation set, and the model is trained on the remaining data points. This process is repeated for each data point. LOOCV is computationally expensive but provides a less biased estimate of model performance.

2. Stratified K-Fold Cross-Validation: This method ensures that each fold contains a proportional representation of each class in the target variable. It is useful for imbalanced datasets and can improve the generalization of the model.

3. Time Series Cross-Validation: For time-series data, where the order of data points is crucial, time series cross-validation methods like Time Series Split or Walk-Forward Validation can be used. These methods preserve the temporal order of data points during the cross-validation process.

4. Group K-Fold Cross-Validation: In cases where data points are grouped or clustered, Group K-Fold Cross-Validation ensures that data points from the same group are kept together in the same fold. This can prevent data leakage and provide a more realistic estimate of model performance.

Each cross-validation method has its strengths and limitations, and the choice of method should be based on the specific characteristics of the dataset and the modeling task at hand.

Question 5) What metrics did you use to evaluate your manual implementations of multivariate regression and k-fold cross-validation, and why did you choose them?

To evaluate the manual implementations of multivariate regression and k-fold cross-validation, the primary metric used was Mean Squared Error (MSE) for the following resaons:.

1. **Sensitivity to Large Errors:** MSE is particularly sensitive to large errors because it squares the differences between predicted and actual values. This characteristic is beneficial in regression analysis as it penalizes significant deviations more heavily, thus pushing the model to minimize large errors and improve overall accuracy.

2. **Differentiability:** The MSE metric is differentiable, which makes it suitable for gradient-based optimization algorithms like gradient descent. The smooth and continuous nature of MSE

Amirreza Akbari | 810899045
Reza Baghestani | 810899046
Hananeh Jamali | 810899053

ensures that the gradient descent algorithm can effectively navigate the error landscape to find the optimal model parameters.

3. **Interpretability:** MSE is easy to interpret as it provides a clear measure of average squared differences between predictions and actual values. A lower MSE indicates a better-fitting model, making it a straightforward and reliable metric for evaluating regression models.

4. **Standard Practice:** MSE is a widely accepted and commonly used metric in the field of regression analysis. Its widespread use facilitates comparison with other models and methods, including those implemented using standard libraries such as scikit-learn.

Application in the Project:

- **Multivariate Regression:** For the custom multivariate regression model, MSE was calculated to assess the model's performance on the test set. This provided a clear indication of how well the model was able to predict the target variables ('Price in Thousands' and 'Horsepower') based on the input features.

- **K-Fold Cross-Validation:** During the manual k-fold cross-validation, MSE was computed for each fold. The average MSE across all folds was then calculated to evaluate the model's generalization ability. This average MSE served as an indicator of how consistently the model performed across different subsets of the data.