Project 1 Report

Machine Learning Regression Analysis

Nicholas Pickering

MSIM607—Machine Learning 1
Fall 2025

Professor: Dr. Li
Old Dominion University

October 25, 2025

Contents

List of Figures

List of Tables

1 Introduction

This project demonstrates machine learning techniques using scikit-learn [4, 5] and evaluates model performance using standard metrics such as mean squared error [8] and coefficient of determination (R^2) [6]. The analysis follows established machine learning practices. For the more complex dataset of the Red Wine Quality, Principal Component Analysis (PCA) [9, 5] was applied for dimensionality reduction to determine the most significant features. Data handling and processing were completed utilizing Polars, which provides the DataFrame capabilities of NumPy and Pandas with enhanced performance for large datasets [2].

2 Salary Dataset

This task involved analyzing a dataset containing salary information for various employees. The objective of this task was to design a linear regression model to fit the data and predict salaries based on years of experience. The model's performance was evaluated using metrics such as Mean Squared Error (MSE) [8], Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-squared (R^2) [6] to assess the accuracy of the predictions.

2.1 Implementation Details

The Linear Regression model was implemented using the `LinearRegression` class from the `sklearn.linear_model` module [4]. The training data was loaded from the provided CSV file using Polars [2]. The features (years of experience) and target variable (salary) were extracted from the dataset and used to fit the `LinearRegression` model.

2.1.1 Data Handling

The dataset was read using Polars as follows:

```
import polars as pl

data_location = 'dataset/salary/Salary_dataset.csv'
data = pl.read_csv(data_location)
X_train = data[['YearsExperience']]
y_train = data['Salary']
```

The resulting dimensions of the feature and target arrays were:

- **X_train**: (30, 1) – 30 samples with 1 feature (Years of Experience)
- **y_train**: (30,) – 30 target values (Salary)

2.1.2 Model Training

The Linear Regression model was trained using the following code:

```
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_train)
```

The resulting dimension of the predictions array was:

- **y_pred**: (30,) – 30 predicted salary values

2.2 Results

2.2.1 Plotted Graph

The resultant graph of the linear regression model is shown in Figure 1.

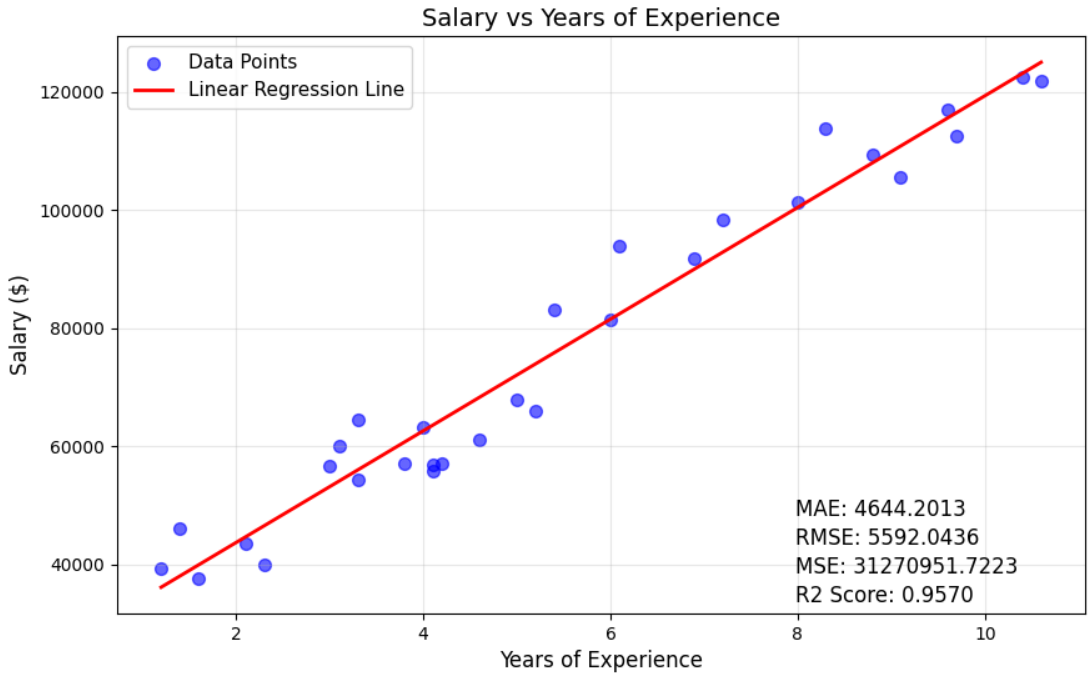

Figure 1: Linear Regression Model for Salary Prediction

2.2.2 Analysis

The performance of the Linear Regression model was evaluated using several metrics. The results are summarized in Table 1.

Metric	Value	Interpretation
R^2 Score	0.9570	Excellent model fit (95.7% variance explained)
MAE	\$4,644.20	Average prediction error of \$4,644
RMSE	\$5,592.04	Root mean squared error of \$5,592
MSE	31,270,952	Mean squared error (in squared dollars)

Table 1: Performance Metrics for Salary Prediction Linear Regression Model

The model demonstrates great performance with an R^2 score of 0.9570, indicating that the linear regression explains 95.7% of the variance in salary based on years of experience. The Mean Absolute Error (MAE) of \$4,644 represents an average prediction error of approximately 5.8% from the regression line to the actual values.

The Root Mean Squared Error (RMSE) of \$5,592 is slightly higher than the MAE, indicating that most predictions are accurate, but there are some instances with larger prediction errors. This can be interpreted as there being a few outliers in the dataset where the actual salary deviates significantly from the predicted salary based on years of experience.

3 Linear Regression Dataset

This task involved analyzing a dataset using linear regression techniques. The objective was to design a linear regression model to fit the data and make predictions based on the input features. This dataset does not have a specified set of features, just a set of x and y train and test values. The efforts in this task reflected the efforts from the Salary Dataset task, but without a specified feature set.

3.1 Implementation Details

The Linear Regression model was implemented using the `LinearRegression` class from the `sklearn.linear_model` module [4]. The training and testing datasets were loaded from the provided CSV files using Polars [2]. The x and y training values were extracted from the dataset and used to fit the `LinearRegression` model. After the training was complete, predictions were made on both the training and testing datasets and statistics were calculated to evaluate the model's performance.

3.1.1 Data Handling

The dataset was read using Polars as follows:

```
import polars as pl

train_data_location = 'dataset/Linear_Regression/train.csv'
test_data_location = 'dataset/Linear_Regression/test.csv'
train_data = pl.read_csv(train_data_location)
test_data = pl.read_csv(test_data_location)
X_train = train_data[['x']]
y_train = train_data['y']
X_test = test_data[['x']]
y_test = test_data['y']
```

The resultant dimensions of the data arrays were:

- **X_train**: (700, 1) – 700 samples with 1 feature (x values for training data)
- **y_train**: (700,) – 700 target values (y values for training data)
- **X_test**: (300, 1) – 300 samples with 1 feature (x values for testing data)
- **y_test**: (300,) – 300 target values (y values for testing data)

3.1.2 Model Training

The Linear Regression model was trained using the following code:

```
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X_train, y_train)
y_train_pred = model.predict(X_train)
y_test_pred = model.predict(X_test)
```

The resultant dimensions of the y_test and y_train prediction arrays were:

- **y_train_pred**: (700,) – 700 predicted y values for training data
- **y_test_pred**: (300,) – 300 predicted y values for testing data

3.2 Results

3.2.1 Plotted Graph

The resultant graphs of the linear regression model comparisons are shown in Figure 2.

Figure 2: Linear Regression Model Plot

3.2.2 Analysis

The performance of the Linear Regression model was evaluated using several metrics. The results are summarized in Table 2.

Metric	Training	Test	Interpretation
R^2 Score	0.9907	0.9888	Excellent fit, minimal overfitting
MSE	7.8888	9.4349	Low error, good generalization
RMSE	2.8087	3.0716	Root mean squared error
MAE	2.2307	2.4158	Average absolute error
Generalization Analysis			
R^2 Difference	0.0019		Model generalizes well

Table 2: Linear Regression Performance Metrics and Generalization Analysis

The small generalization gap [7] between the training and test R^2 scores (0.0019) indicates that the model generalizes well to unseen data, with minimal overfitting. The low MSE, RMSE, and MAE values when compared to the data ranges reinforce the generalization analysis for the model's accuracy in predicting the target variable.

4 Red Wine Quality Dataset

This task involved analyzing a dataset containing various chemical properties of red wine samples along with their quality ratings. The objective of this task was to design both a **Linear** and a **Logistic** Regression model to predict wine quality based on the chemical features. Additionally, Principal Component Analysis (PCA) [9] was performed to identify the most significant features contributing to wine quality.

4.1 Implementation Details

4.1.1 Data Handling

The dataset was read using Polars as follows:

```
import polars as pl

data_location = 'dataset/wine_quality/winequality-red.csv'
data = pl.read_csv(data_location)
x_train = data.select(pl.exclude('quality'))
y_train = data['quality']
```

The resultant dimensions of the data array was:

- **data**: (1599, 12) – 1599 samples with 12 features (x values for training data)

After training the Linear and Logistic Regression models, the data was centered to prepare for PCA [9]. The method for centering the data was to subtract the column-wise mean from each value in the respective column. Using Polars, this is a bit more involved than with Pandas, but can be accomplished as shown below. The data centering code was as follows:

```
train_mean = x_train.mean()

x_train_centered = x_train.with_columns(
    [(pl.col(col) - train_mean[col]) for col in x_train.columns]
)
pca = PCA(n_components=2)
x_train_reduced = pca.fit_transform(x_train_centered)
```

4.1.2 Model Training

The Linear Regression model was trained using the following code:

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score

linear_model = LinearRegression()
cross_validation_score = cross_val_score(linear_model, x_train, y_train, cv=3,
↪  error_score='raise')
```

4.1.3 Cross Validation and Binary Classification

The binary classification was completed using the following code:

```
import numpy as np

binary_classification = np.where(y_train > 5, 'High', 'Low')
```

Three-fold cross validation was performed using the cross_val_score function from sklearn.model_selection [3]. Binary classification was completed using the `np.where()` function from NumPy [1] to classify wine quality values greater than 5 as "High", all others as "Low".

The resultant dimension of the validation and classification arrays were:

- **cross_validation_score**: (3,) – 3 cross validation scores for the 3 folds
- **binary_classification**: (1599,) – 1599 binary classification labels

4.2 Results

4.2.1 Plotted Graph

The resultant graphs of the linear regression model comparisons are shown in Figure 3.

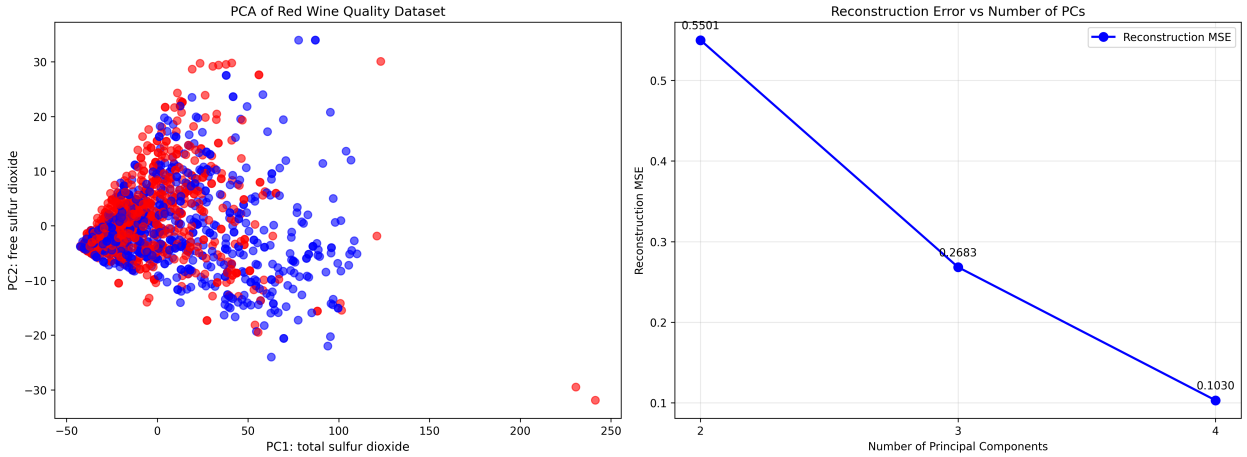

Figure 3: Red Wine Quality Linear Regression Model Plot

4.2.2 Tabular Outputs

The results of the **Linear** Regression model are summarized in Table 3.

Metric	Value	Interpretation
Cross Validation Scores	[0.270419 0.354977 0.309328]	R^2 scores for each fold
Mean Cross Validation Score	0.311	Average R^2 score

Table 3: Red Wine Quality Linear Regression Model Results

The results of the **Logistic** Regression model are summarized in Table 4.

Class	Count	Interpretation
Cross Validation Scores	[0.705441 0.733583 0.748593]	R^2 scores for each fold
Mean Cross Validation Score	0.729	Average R^2 score

Table 4: Red Wine Quality Logistic Regression Model Results

The PCA results are summarized in Table 5.

Component	Variance Explained	Top Contributing Features
PC1	94.66%	Total sulfur dioxide (0.976) Free sulfur dioxide (0.219)
PC2	4.84%	Free sulfur dioxide (0.975) Total sulfur dioxide (-0.219)
Combined PC1+PC2	99.49%	Sulfur dioxide compounds dominate variance

Table 5: PCA Results for Wine Quality Dataset

The results of using different numbers of principal components is shown in Table 6.

Components	MSE	Explained Variance	Reconstruction Error
2 PCs	0.550	99.49%	0.51%
3 PCs	0.268	99.75%	0.25%
4 PCs	0.103	99.91%	0.09%

Table 6: PCA Results Using Different Numbers of Principal Components

4.2.3 Analysis

The Red Wine Quality dataset was analyzed using Linear Regression, Logistic Regression, and Principal Component Analysis (PCA). Both the Linear and Logistic Regression models were evaluated using a three-fold cross-validation approach to determine the model's performance. The mean R^2 score across the three folds of the linear model was approximately 0.311, indicating a moderate fit of this model to the data. This explains that the linear model is only able to account for ˜31.1% of the variance in wine quality based on the provided dataset.

The Logistic Regression model, which utilized binary classification of wine quality into "High" and "Low" categories, resulted in a mean R^2 score of approximately 0.729 across the three folds. This indicates that, for this dataset, the logistic model is a better fit than the linear model, explaining ˜72.9% of the variance in wine quality. This suggests that the relationship between the features and wine quality is better captured by a classification approach rather than a regression approach.

PCA was performed to determine which features contributed most to the variance in the dataset. The first two principal components (PC1 and PC2) explained a combined 99.49% of the variance, with PC1 alone accounting for 94.66%. The top contributing features to PC1 were total sulfur dioxide and free sulfur dioxide, indicating that these chemicals are significant factors in determining wine quality. Using increasing numbers of principal components showed that even with just 2 PCs, the reconstruction error was only 0.51%, indicating that the majority of the information in the dataset can be captured with a very low-dimensional representation (Total Sulfur Dioxide, Free Sulfur Dioxide, Resultant Wine Quality). Intuition would tell that wine quality is influenced by a combination of factors, not just these two features alone. However, PCA identifies the directions of maximum variance in the data, which may not directly correspond to the most intuitive or expected features.

References

[1] NumPy developers. numpy.where — numpy v2.3 manual. https://numpy.org/doc/2.3/reference/generated/numpy.where.html, 2025. Accessed: 2025-10-14.

[2] Polars developers. Polars documentation. https://docs.pola.rs/, 2025. Accessed: 2025-10-14.

[3] Scikit-learn developers. Cross-validation: evaluating estimator performance. https://scikit-learn.org/stable/modules/cross_validation.html, 2025. Accessed: 2025-10-14.

[4] Scikit-learn developers. Linear models. https://scikit-learn.org/stable/modules/linear_model.html, 2025. Accessed: 2025-10-14.

[5] Scikit-learn developers. sklearn.decomposition.pca. https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html, 2025. Accessed: 2025-10-14.

[6] Wikipedia contributors. Coefficient of determination — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Coefficient_of_determination, 2025. Accessed: 2025-10-14.

[7] Wikipedia contributors. Generalization error — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Generalization_error, 2025. Accessed: 2025-10-14.

[8] Wikipedia contributors. Mean squared error — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Mean_squared_error, 2025. Accessed: 2025-10-14.

[9] Wikipedia contributors. Principal component analysis — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Principal_component_analysis, 2025. Accessed: 2025-10-14.

[10] Wikipedia contributors. Scoring rule — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Scoring_rule, 2025. Accessed: 2025-10-14.