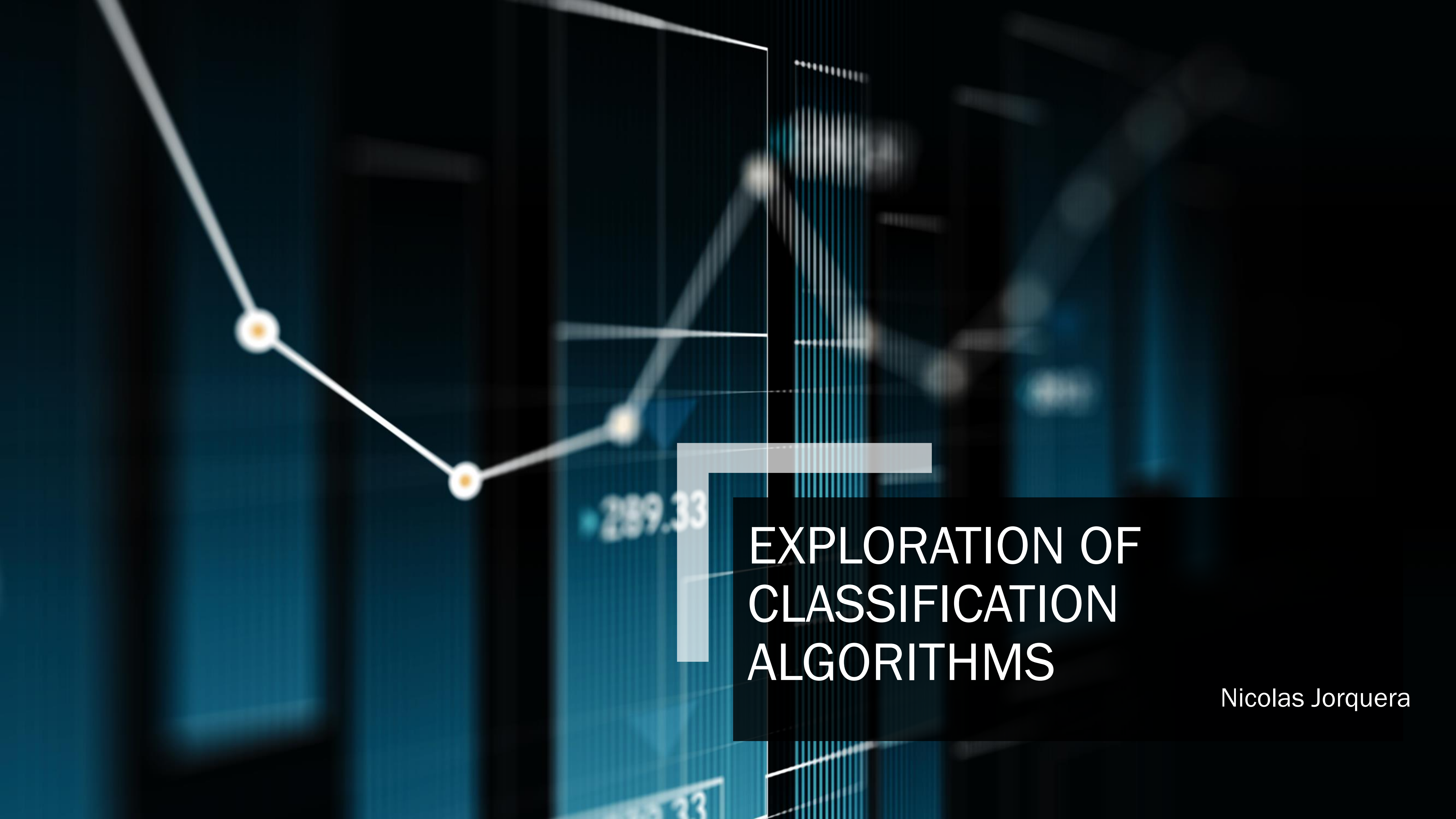
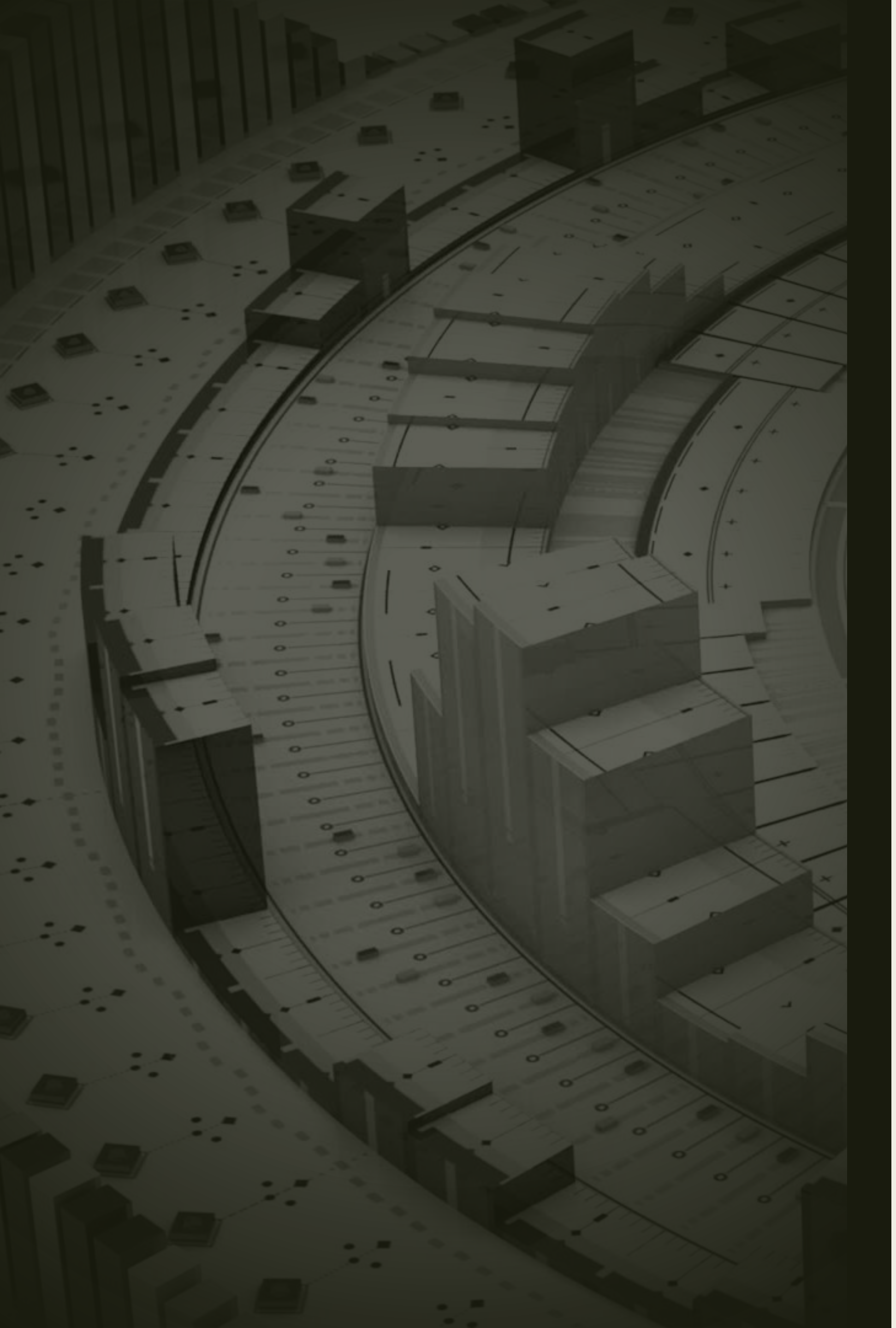


EXPLORATION OF CLASSIFICATION ALGORITHMS

Nicolas Jorquera

Problem Statement

- In the world of finance, credit scores play a crucial role in determining an individual’s creditworthiness.
- Accurate credit score classification is essential for financial institutions to make informed decisions regarding loans, interest rates, and credit limits
- In this project, I aim to develop a machine learning model that can effectively classify credit scores into three categories: Good, Poor, and Standard.

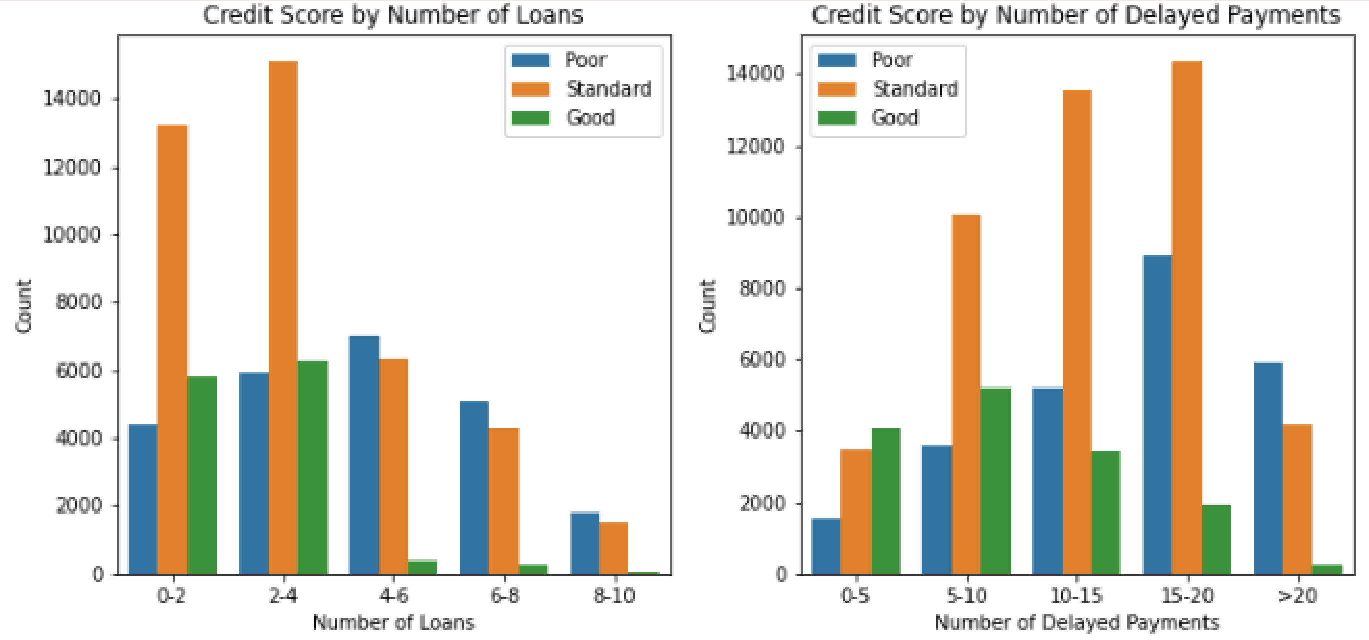


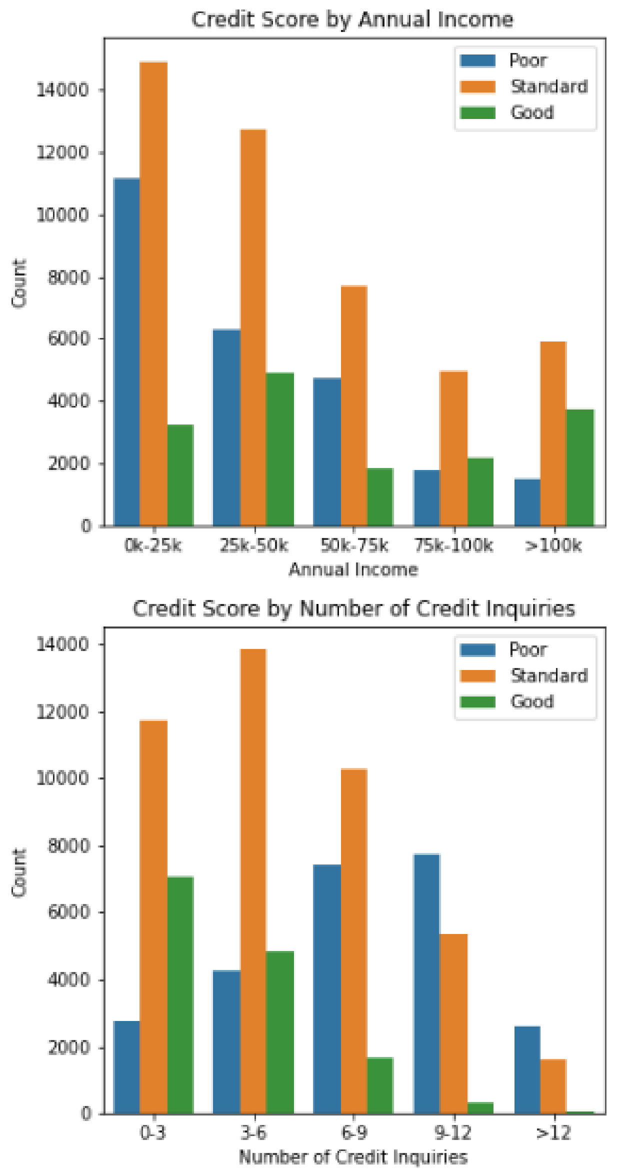
Data Set Description

- The data set used for this project consists of 100,000 records with 28 features, including demographic information, financial behavior, and credit history.
- The target variable is the credit score, which has three possible values: Good, Poor, and Standard
- The objective is to build a machine learning model that can predict the credit score category based on the given features.

Data Visualization

- To better understand the relationships between different features and credit scores, I created a series of visualizations. I grouped the data into various categories based on age, annual income, monthly balance, outstanding debt, number of credit inquiries, number of bank accounts, number of loans, and number of delayed payments.





Credit Score by Annual Income

Credit Score by Number of Credit Inquiries

Data Visualization

- The distribution of credit scores for most categories appeared to be normal.
- Poor credit scores were more common among certain age groups, those with higher outstanding debt, higher number of loans and high delayed payments. These factors intuitively make sense since young people tend to have less financial stability, and loans / delayed payments have a direct impact on credit score.
- High credit scores were common among high income individuals with higher income, and these individuals have minimal to no loans, credit inquiries and outstanding deb

Machine Learning Algorithms

Decision Tree Classifier

Random Forest Classifier

Gradient Boosting Classifier

Implementation Results

Data Preprocessing

Train – Test – Split

Model Training and Validation

Hyperparameter Tuning

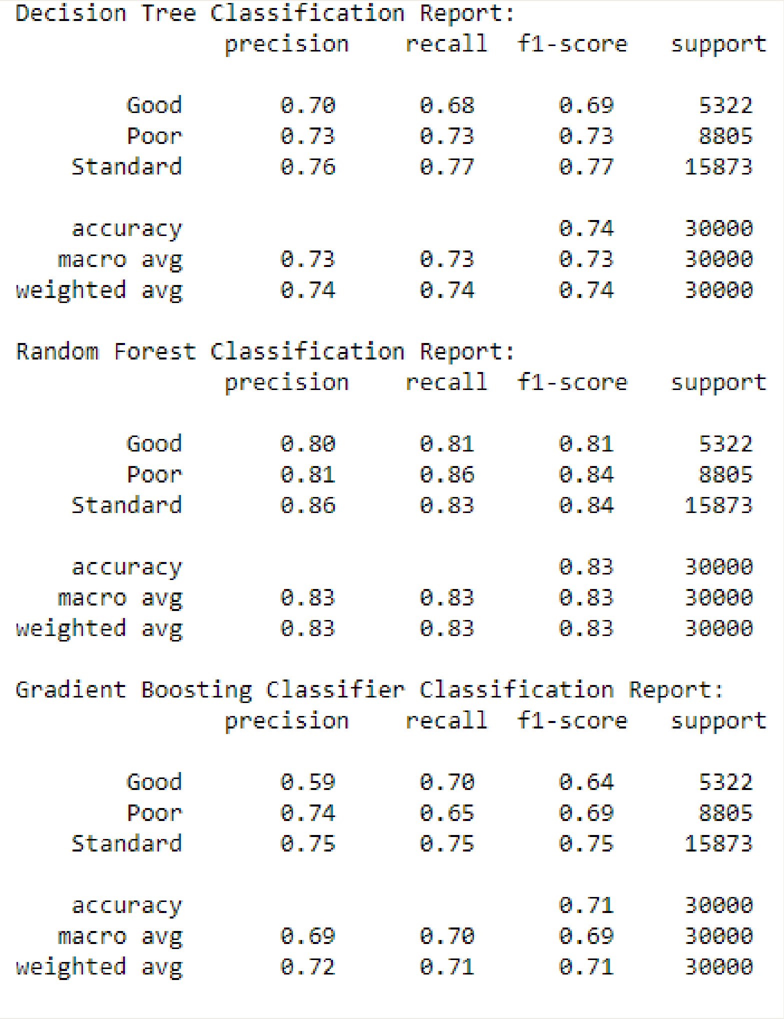
Model Selection

Performance Metrics

PRECISION

RECALL

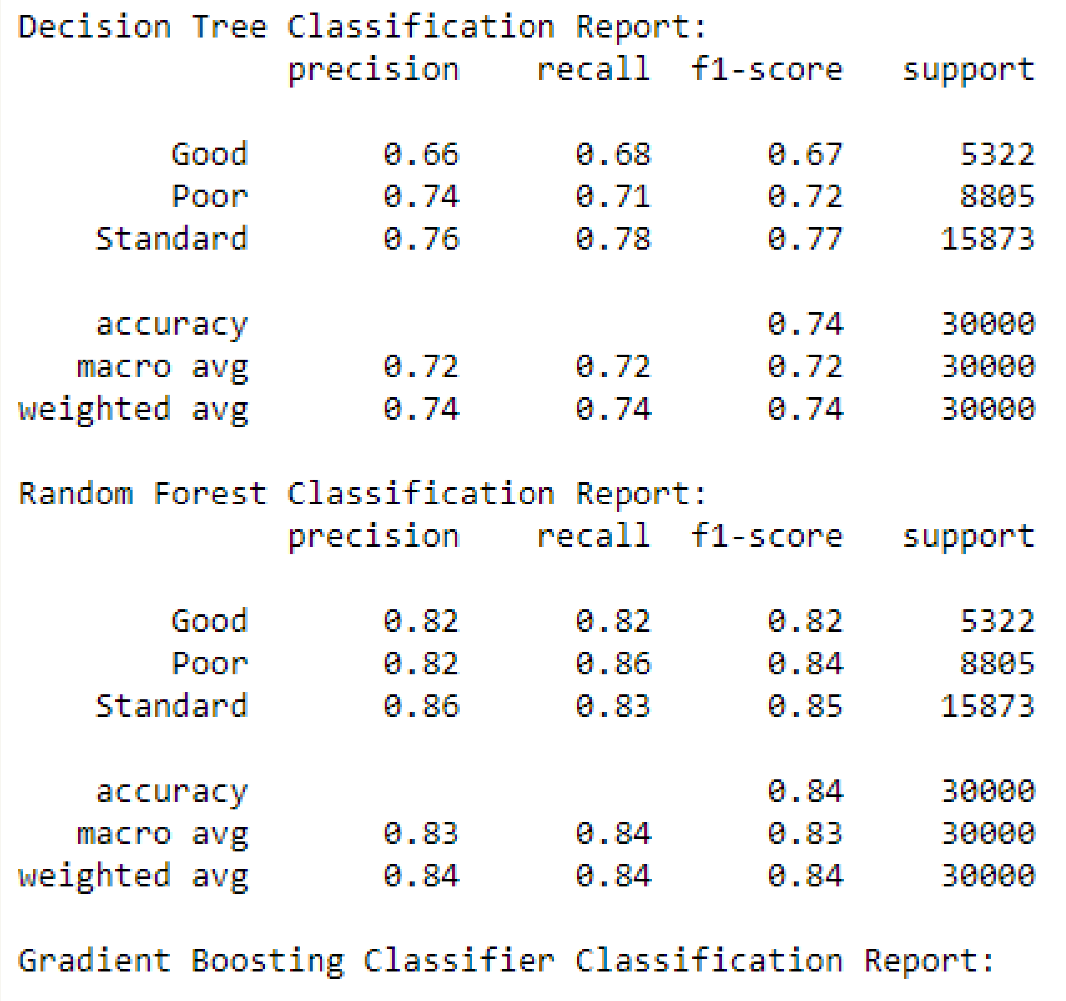
F1 - SCORE



```
Decision Tree Classification Report:
              precision    recall  f1-score   support

        Good       0.70      0.68      0.69      5322
        Poor       0.73      0.73      0.73      8805
    Standard       0.76      0.77      0.77     15873

    accuracy                           0.74     30000
   macro avg       0.73      0.73      0.73     30000
weighted avg       0.74      0.74      0.74     30000

Random Forest Classification Report:
              precision    recall  f1-score   support

        Good       0.80      0.81      0.81      5322
        Poor       0.81      0.86      0.84      8805
    Standard       0.86      0.83      0.84     15873

    accuracy                           0.83     30000
   macro avg       0.83      0.83      0.83     30000
weighted avg       0.83      0.83      0.83     30000

Gradient Boosting Classifier Classification Report:
              precision    recall  f1-score   support

        Good       0.59      0.70      0.64      5322
        Poor       0.74      0.65      0.69      8805
    Standard       0.75      0.75      0.75     15873

    accuracy                           0.71     30000
   macro avg       0.69      0.70      0.69     30000
weighted avg       0.72      0.71      0.71     30000
```

Comparison

- Random Forests perform the best in this problem because they effectively mitigate overfitting and handle noise better by averaging the predictions of multiple decision trees, leading to a more accurate and stable model.
- Gradient Boosting Classifier can also perform well when tuned correctly, but it might be more sensitive to noise and overfitting depending on the learning rate and number of boosting iterations. This is proved as both Gradient Boosting Classifier and Random Forest perform significantly better than decision trees.
- Decision Trees, while simple and easy to understand, tend to suffer from overfitting and are less robust to noise compared to the ensemble methods.



```
Decision Tree Classification Report:
              precision    recall  f1-score   support

        Good       0.66      0.68      0.67      5322
        Poor       0.74      0.71      0.72      8805
    Standard       0.76      0.78      0.77     15873

    accuracy                           0.74     30000
   macro avg       0.72      0.72      0.72     30000
weighted avg       0.74      0.74      0.74     30000

Random Forest Classification Report:
              precision    recall  f1-score   support

        Good       0.82      0.82      0.82      5322
        Poor       0.82      0.86      0.84      8805
    Standard       0.86      0.83      0.85     15873

    accuracy                           0.84     30000
   macro avg       0.83      0.84      0.83     30000
weighted avg       0.84      0.84      0.84     30000

Gradient Boosting Classifier Classification Report:
```

Improving Model Performance – Randomized Search CV

- As expected, Random Forest improved slightly, however not as significant as Decision Trees. This is most likely because the Random Forest default parameters were most likely close to the parameters found using RandomSearchCV; which signifies the model was almost optimized.
- To find a more optimal solution I can use GridSearchCV but I would not expect to see significant improvements in performance. Alternatively, Decision Tree was not optimized therefore the accuracy improved by almost 20 percent.

Future Improvements

- Neural Networks
- Professional Expertise
- Regression Model

Conclusion

- In this project, we aimed to classify credit scores into three categories: Good, Poor, and Standard.
- We started by implementing three classification algorithms: Decision Tree, Random Forest, and Gradient Boosting Classifier.
- We evaluated their performance using classification reports, which provided precision, recall, f1-score, and accuracy metrics for each algorithm.
- Next, we used RandomizedSearchCV to perform hyperparameter tuning on these three algorithms, aiming to improve their performance.
- We compared their performance after tuning and observed improvements in the metrics for each algorithm compared to their default parameters