AI PROJECT

CREDIT CARD APPROVAL

Project Report

TABLE OF CONTENTS

Name	ID
Ammar Alaa	2022/01210
Adham Mohamed	2022/04058
Youssef Abdelshahid	2022/05890
Youssef Amer	2022/07525
Abdelrhman Mohsen	2022/02140

INTRODUCTION

- **Objective:** To predict credit card approval status (Approved/Not Approved) using classification techniques.
- **Overview:** Our Datasets involve the data of many clients like their Income, Date of Birth, Occupation, Education level, Relationship Status, Housing Status, etc.

Credit Scores to Predict

Approved Status

- **'C': Likely denotes a "Credit Approved" status.**
- **'X': Possibly stands for "Accepted" or "Approved." These statuses indicate that the application was successful, so they are grouped under the same category.**

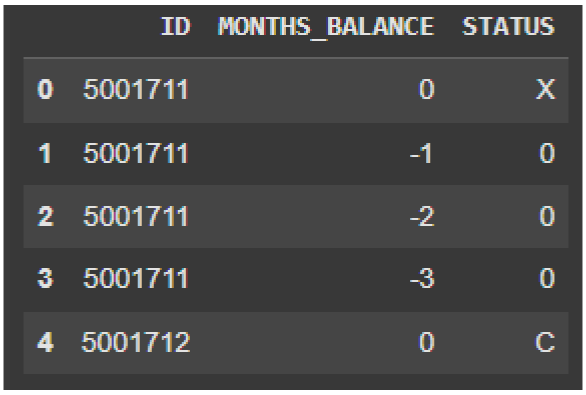
Not Approved Status

- **'0': Application not approved, but no specific issue provided.**
- **'1': Denial due to minor reasons, such as insufficient credit history.**
- **'2': Denial due to more severe issues, such as a high debt-to-income ratio.**
- **'3', '4', '5': Represent escalating levels of rejection or issues, such as bankruptcy, delinquency, or fraud.**

1. DATA EXPLORATION AND STATISTICAL ANALYSIS

Dataset Description:

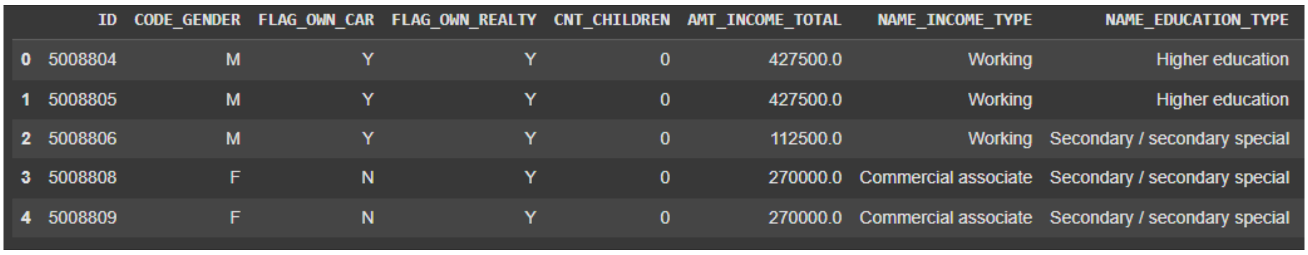
- Provide details about the credit_record and application_record datasets.
    - credit_record dataset contained the following attributes:



	ID	MONTHS_BALANCE	STATUS
0	5001711	0	X
1	5001711	-1	0
2	5001711	-2	0
3	5001711	-3	0
4	5001712	0	C

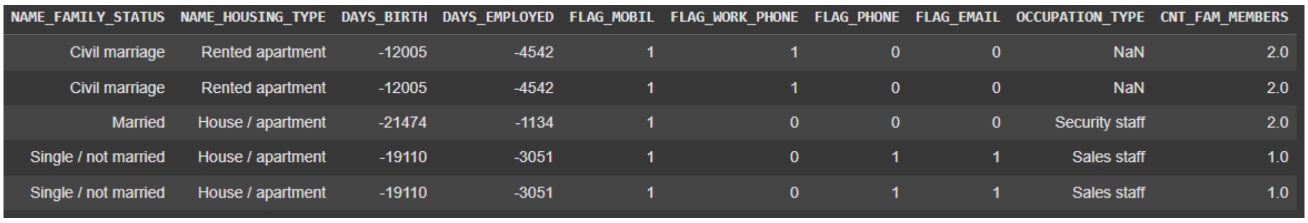
Figure 1: credit_record dataset

- application_record dataset contained the following attributes:



	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_INCOME_TYPE	NAME_EDUCATION_TYPE
0	5008804	M	Y	Y	0	427500.0	Working	Higher education
1	5008805	M	Y	Y	0	427500.0	Working	Higher education
2	5008806	M	Y	Y	0	112500.0	Working	Secondary / secondary special
3	5008808	F	N	Y	0	270000.0	Commercial associate	Secondary / secondary special
4	5008809	F	N	Y	0	270000.0	Commercial associate	Secondary / secondary special

Figure 2: application_record dataset



NAME_FAMILY_STATUS	NAME_HOUSING_TYPE	DAYS_BIRTH	DAYS_EMPLOYED	FLAG_MOBIL	FLAG_WORK_PHONE	FLAG_PHONE	FLAG_EMAIL	OCCUPATION_TYPE	CNT_FAM_MEMBERS
Civil marriage	Rented apartment	-12005	-4542	1	1	0	0	NaN	2.0
Civil marriage	Rented apartment	-12005	-4542	1	1	0	0	NaN	2.0
Married	House / apartment	-21474	-1134	1	0	0	0	Security staff	2.0
Single / not married	House / apartment	-19110	-3051	1	0	1	1	Sales staff	1.0
Single / not married	House / apartment	-19110	-3051	1	0	1	1	Sales staff	1.0

Figure 3: application_record dataset continued

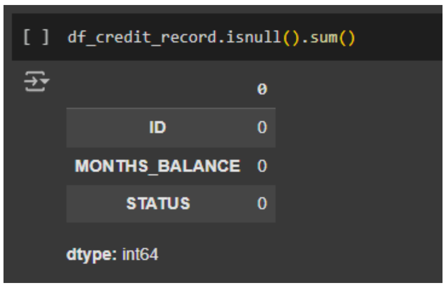
EDA (Exploratory Data Analysis):

We found that there were missing values in the coloum “Occupation Title” and there were duplicates in both datasets which needed to be dealt with.

2. DATA PREPROCESSING

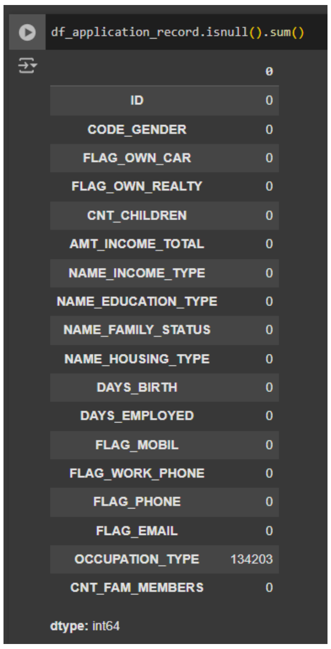
Steps Taken:

- We Addressed the null and duplicate values by firstly identifying them using isnull() function.



```
[ ] df_credit_record.isnull().sum()

                      0
              ID      0
  MONTHS_BALANCE      0
          STATUS      0

dtype: int64
```

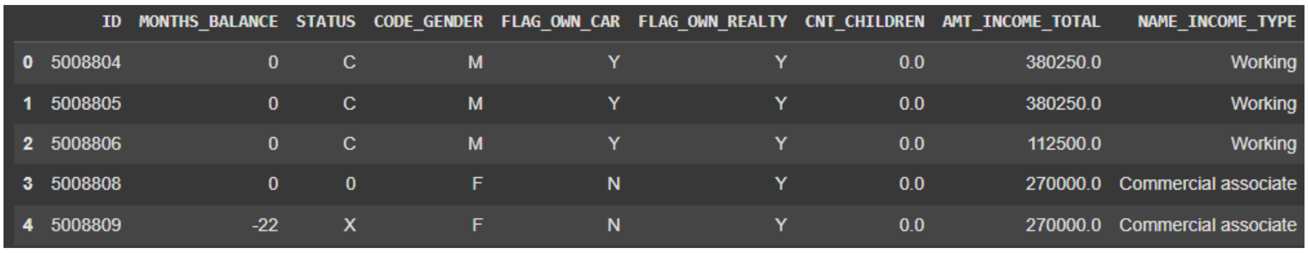
Figure 4: Identifying Null Values (credit_record)



```
df_application_record.isnull().sum()

                              0
                   ID         0
          CODE_GENDER         0
         FLAG_OWN_CAR         0
      FLAG_OWN_REALTY         0
         CNT_CHILDREN         0
     AMT_INCOME_TOTAL         0
     NAME_INCOME_TYPE         0
  NAME_EDUCATION_TYPE         0
   NAME_FAMILY_STATUS         0
    NAME_HOUSING_TYPE         0
           DAYS_BIRTH         0
        DAYS_EMPLOYED         0
           FLAG_MOBIL         0
      FLAG_WORK_PHONE         0
           FLAG_PHONE         0
           FLAG_EMAIL         0
      OCCUPATION_TYPE    134203
      CNT_FAM_MEMBERS         0

dtype: int64
```

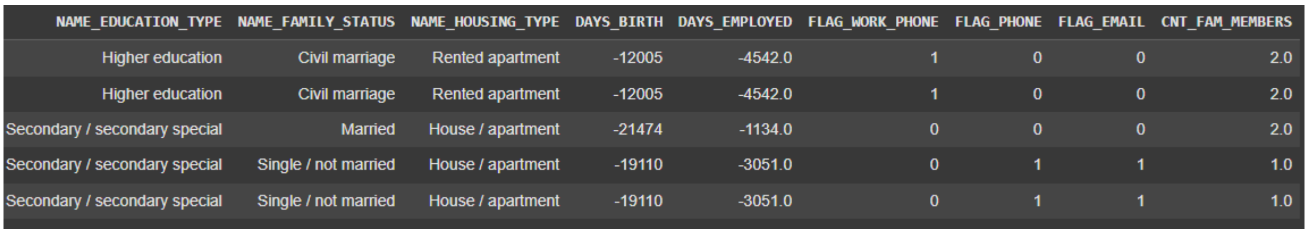
Figure 5: Identifying Null Values (application_record) (30.6%)

- The Merge was done based on ID



	ID	MONTHS_BALANCE	STATUS	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_INCOME_TYPE
0	5008804	0	C	M	Y	Y	0.0	380250.0	Working
1	5008805	0	C	M	Y	Y	0.0	380250.0	Working
2	5008806	0	C	M	Y	Y	0.0	112500.0	Working
3	5008808	0	0	F	N	Y	0.0	270000.0	Commercial associate
4	5008809	-22	X	F	N	Y	0.0	270000.0	Commercial associate

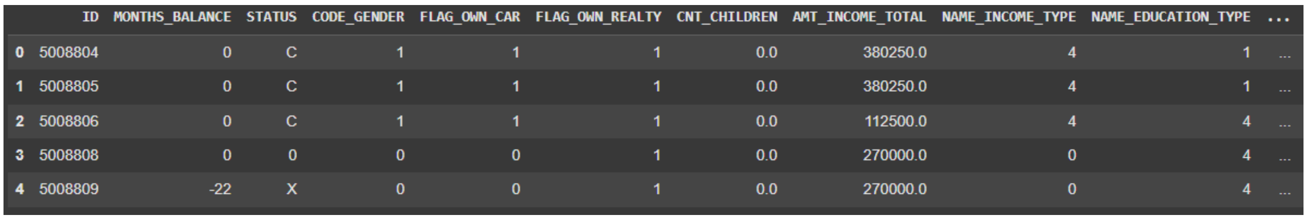
Figure 6: Merged Dataset Based On ID



NAME_EDUCATION_TYPE	NAME_FAMILY_STATUS	NAME_HOUSING_TYPE	DAYS_BIRTH	DAYS_EMPLOYED	FLAG_WORK_PHONE	FLAG_PHONE	FLAG_EMAIL	CNT_FAM_MEMBERS
Higher education	Civil marriage	Rented apartment	-12005	-4542.0	1	0	0	2.0
Higher education	Civil marriage	Rented apartment	-12005	-4542.0	1	0	0	2.0
Secondary / secondary special	Married	House / apartment	-21474	-1134.0	0	0	0	2.0
Secondary / secondary special	Single / not married	House / apartment	-19110	-3051.0	0	1	1	1.0
Secondary / secondary special	Single / not married	House / apartment	-19110	-3051.0	0	1	1	1.0

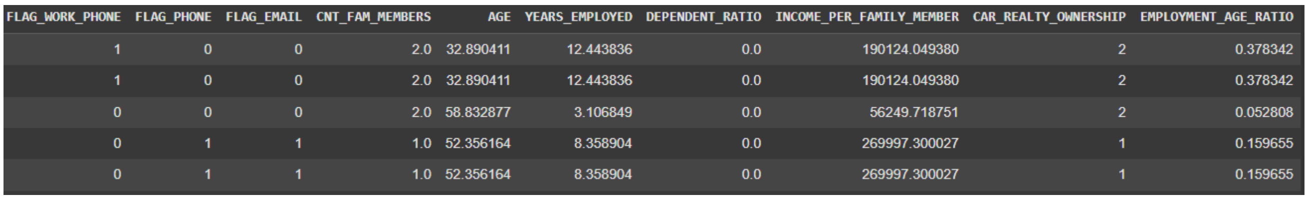
Figure 7: Merged Dataset Based On ID Continued

- Regarding Scaling, we have done it to fix continuous values like the month balance for example.
- Regarding label encoding, we converted categorical columns to numerical values to help the models and feature scaling function better.
- Regarding SMOTE and SMOTEENN, this after implementation, helped dramatically to fix class imbalance.
- Feature Engineering helped us simplify our data for further optimization to help achieve better accuracy.



	ID	MONTHS_BALANCE	STATUS	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_INCOME_TYPE	NAME_EDUCATION_TYPE	...
0	5008804	0	C	1	1	1	0.0	380250.0	4	1	...
1	5008805	0	C	1	1	1	0.0	380250.0	4	1	...
2	5008806	0	C	1	1	1	0.0	112500.0	4	4	...
3	5008808	0	0	0	0	1	0.0	270000.0	0	4	...
4	5008809	-22	X	0	0	1	0.0	270000.0	0	4	...

Figure 8: Feature Engineering Results



FLAG_WORK_PHONE	FLAG_PHONE	FLAG_EMAIL	CNT_FAM_MEMBERS	AGE	YEARS_EMPLOYED	DEPENDENT_RATIO	INCOME_PER_FAMILY_MEMBER	CAR_REALTY_OWNERSHIP	EMPLOYMENT_AGE_RATIO
1	0	0	2.0	32.890411	12.443836	0.0	190124.049380	2	0.378342
1	0	0	2.0	32.890411	12.443836	0.0	190124.049380	2	0.378342
0	0	0	2.0	58.832877	3.106849	0.0	56249.718751	2	0.052808
0	1	1	1.0	52.356164	8.358904	0.0	269997.300027	1	0.159655
0	1	1	1.0	52.356164	8.358904	0.0	269997.300027	1	0.159655

Figure 9: Feature Engineering Results Continued

- The Heatmap allowed us to visualize the correlation between every column and status column to see the effect of each column on the final classification/output.

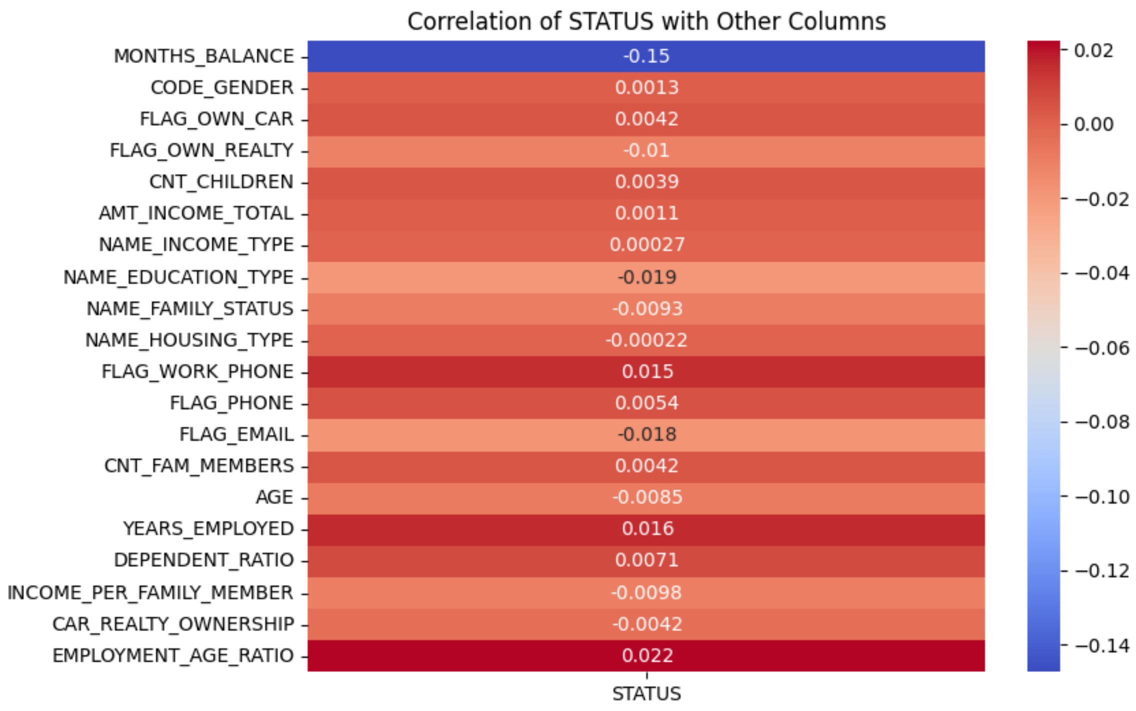


Figure 10: Heatmap

3. FEATURE SELECTION

Genetic Algorithm Approach:

In this project, GAs are applied to feature selection, aiming to identify the most important features that maximize the model's predictive performance. This helps reduce dimensionality, improve model interpretability, and enhance computational efficiency.

Process of Feature Selection Using Genetic Algorithms

- **Initialization of Population**:
  - A population of chromosomes is generated, where each chromosome represents a subset of features.
  - A chromosome is a binary array (True for selected features, False otherwise), with a mix of included and excluded features initialized randomly.
- **Fitness Evaluation**:
  - The fitness of each chromosome is measured by training the model (e.g., DecisionTreeClassifier) on the selected features and evaluating its accuracy on the validation set.
  - The fitness_score function ranks chromosomes based on their validation accuracy.
- **Selection of Elites and Parents**:
  - **Elite Selection**: Top-performing chromosomes are retained to ensure the best solutions persist across generations.
  - **Parent Selection**: A subset of top chromosomes is chosen for producing offspring via crossover.

- **Crossover**:
    - Chromosomes are combined to create new offspring by swapping segments between two parents at a crossover point. This introduces diversity while preserving strong traits.
- **Mutation**:
    - Adaptive mutation modifies certain genes in chromosomes based on the mutation rate and generation. This prevents premature convergence and explores alternative solutions.
- **Generations**:
    - The GA iteratively evolves the population for a fixed number of generations. At each step, the best-performing chromosomes and scores are tracked.

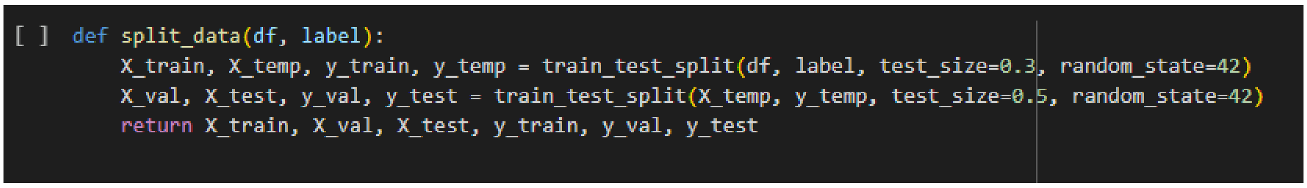
Results of Feature Selection

- **Selected Features**:
  After running the GA, the features selected by the best-performing chromosome are used for model training. These features are deemed most significant for predicting credit card approval status.
- **Importance to Models**:
  The selected features improve model accuracy and reduce overfitting by eliminating irrelevant or redundant data. Additionally, they speed up training and evaluation processes.

4. DATA SPLITTING

Procedure:

- We split the dataset into training (70%), validation (15%), and testing (15%).



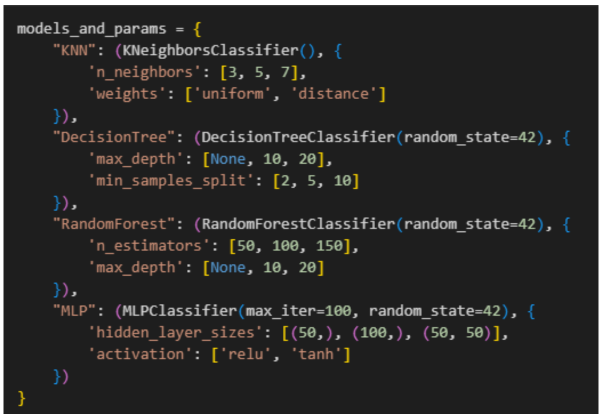
```
[ ]  def split_data(df, label):
         X_train, X_temp, y_train, y_temp = train_test_split(df, label, test_size=0.3, random_state=42)
         X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)
         return X_train, X_val, X_test, y_train, y_val, y_test
```

Figure 11: Data Splitting

5. MODEL SELECTION AND TRAINING

Algorithms Used:

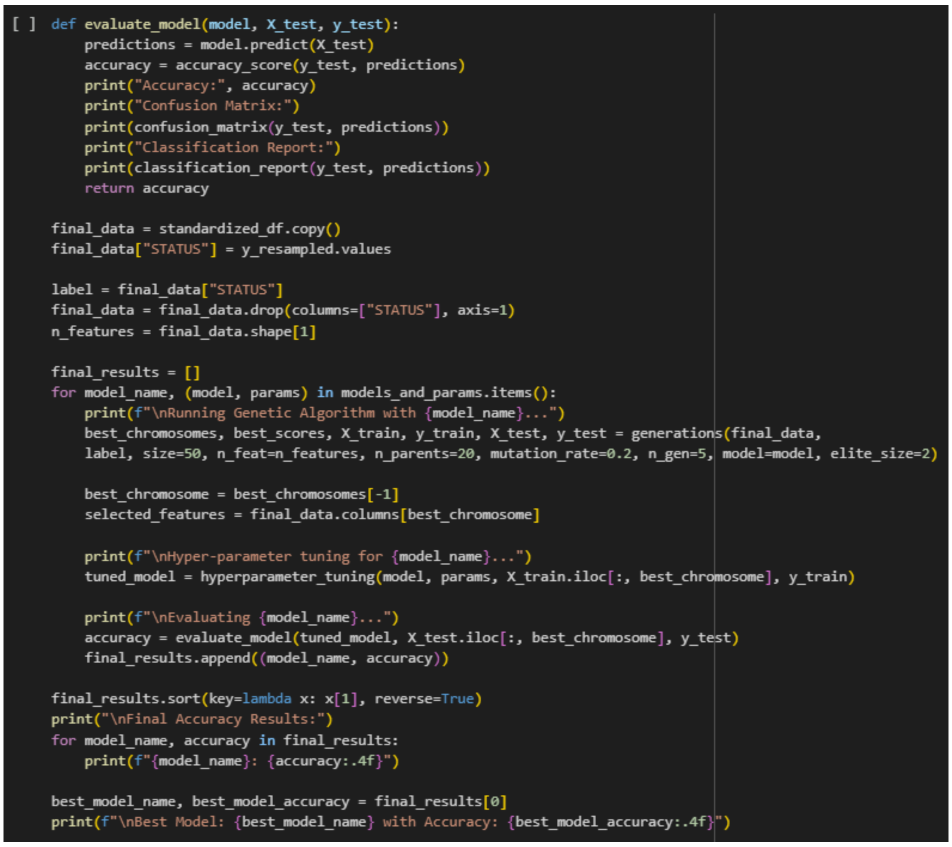
- KNN, Decision Trees, Random Forest, and MLP were used.



```
models_and_params = {
    "KNN": (KNeighborsClassifier(), {
        'n_neighbors': [3, 5, 7],
        'weights': ['uniform', 'distance']
    }),
    "DecisionTree": (DecisionTreeClassifier(random_state=42), {
        'max_depth': [None, 10, 20],
        'min_samples_split': [2, 5, 10]
    }),
    "RandomForest": (RandomForestClassifier(random_state=42), {
        'n_estimators': [50, 100, 150],
        'max_depth': [None, 10, 20]
    }),
    "MLP": (MLPClassifier(max_iter=100, random_state=42), {
        'hidden_layer_sizes': [(50,), (100,), (50, 50)],
        'activation': ['relu', 'tanh']
    })
}
```

Figure 12: Model Selection

- Each Model was trained using the genetic algorithm explained above.

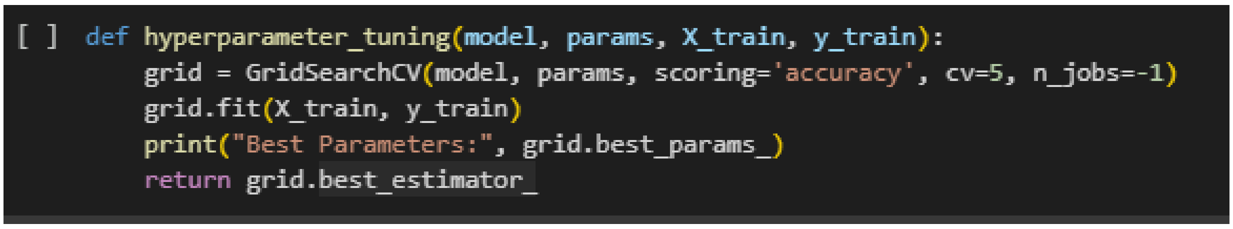


```
[ ] def evaluate_model(model, X_test, y_test):
        predictions = model.predict(X_test)
        accuracy = accuracy_score(y_test, predictions)
        print("Accuracy:", accuracy)
        print("Confusion Matrix:")
        print(confusion_matrix(y_test, predictions))
        print("Classification Report:")
        print(classification_report(y_test, predictions))
        return accuracy

    final_data = standardized_df.copy()
    final_data["STATUS"] = y_resampled.values

    label = final_data["STATUS"]
    final_data = final_data.drop(columns=["STATUS"], axis=1)
    n_features = final_data.shape[1]

    final_results = []
    for model_name, (model, params) in models_and_params.items():
        print(f"\nRunning Genetic Algorithm with {model_name}...")
        best_chromosomes, best_scores, X_train, y_train, X_test, y_test = generations(final_data,
        label, size=50, n_feat=n_features, n_parents=20, mutation_rate=0.2, n_gen=5, model=model, elite_size=2)

        best_chromosome = best_chromosomes[-1]
        selected_features = final_data.columns[best_chromosome]

        print(f"\nHyper-parameter tuning for {model_name}...")
        tuned_model = hyperparameter_tuning(model, params, X_train.iloc[:, best_chromosome], y_train)

        print(f"\nEvaluating {model_name}...")
        accuracy = evaluate_model(tuned_model, X_test.iloc[:, best_chromosome], y_test)
        final_results.append((model_name, accuracy))

    final_results.sort(key=lambda x: x[1], reverse=True)
    print("\nFinal Accuracy Results:")
    for model_name, accuracy in final_results:
        print(f"{model_name}: {accuracy:.4f}")

    best_model_name, best_model_accuracy = final_results[0]
    print(f"\nBest Model: {best_model_name} with Accuracy: {best_model_accuracy:.4f}")
```

Figure 13: Model Training

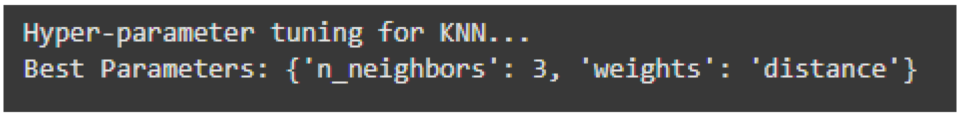
6. HYPERPARAMETER TUNING

- **Approach**:
  - We used grid-search for each model.
- **Results**:



```
[ ]  def hyperparameter_tuning(model, params, X_train, y_train):
         grid = GridSearchCV(model, params, scoring='accuracy', cv=5, n_jobs=-1)
         grid.fit(X_train, y_train)
         print("Best Parameters:", grid.best_params_)
         return grid.best_estimator_
```

Figure 14: Hyperparameter Tuning



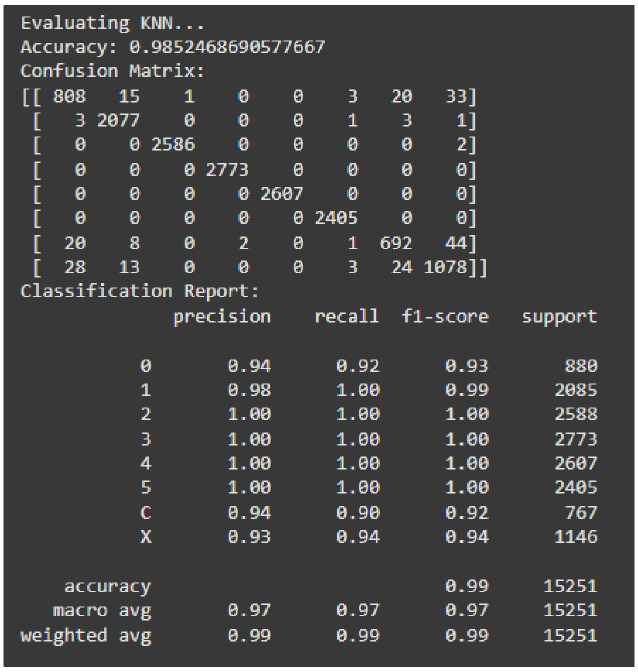
```
Hyper-parameter tuning for KNN...
Best Parameters: {'n_neighbors': 3, 'weights': 'distance'}
```

Figure 15: Hyperparameter Tuning Output

7. MODEL EVALUATION

Metrics:

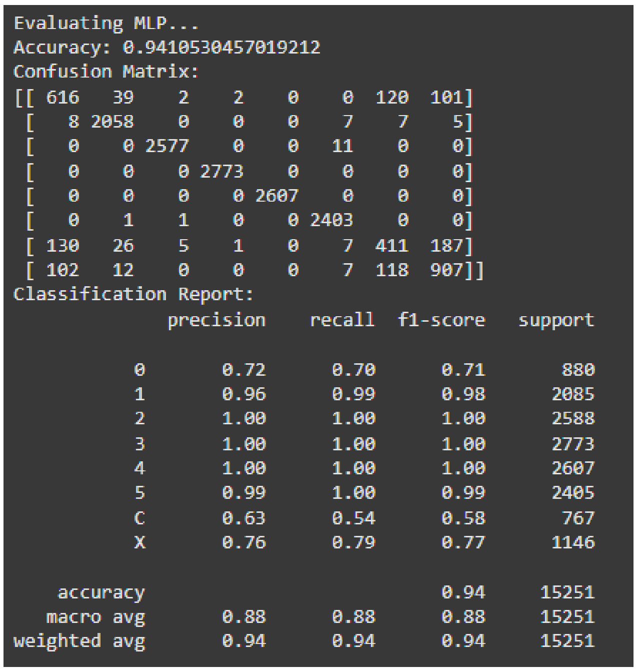
- KNN:



```
Evaluating KNN...
Accuracy: 0.9852468690577667
Confusion Matrix:
[[ 808   15    1    0    0    3   20   33]
 [    3 2077    0    0    0    1    3    1]
 [    0    0 2586    0    0    0    0    2]
 [    0    0    0 2773    0    0    0    0]
 [    0    0    0    0 2607    0    0    0]
 [    0    0    0    0    0 2405    0    0]
 [   20    8    0    2    0    1  692   44]
 [   28   13    0    0    0    3   24 1078]]
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.92      0.93       880
           1       0.98      1.00      0.99      2085
           2       1.00      1.00      1.00      2588
           3       1.00      1.00      1.00      2773
           4       1.00      1.00      1.00      2607
           5       1.00      1.00      1.00      2405
           C       0.94      0.90      0.92       767
           X       0.93      0.94      0.94      1146

    accuracy                           0.99     15251
   macro avg       0.97      0.97      0.97     15251
weighted avg       0.99      0.99      0.99     15251
```

Figure 16: KNN

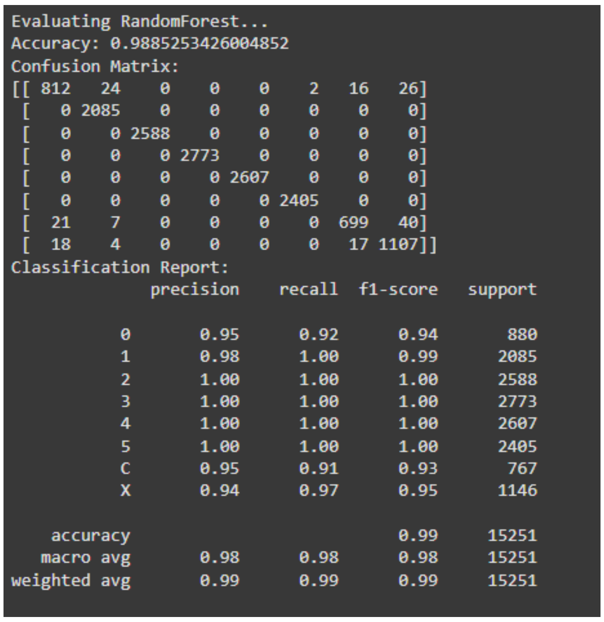
- MLP:



```
Evaluating MLP...
Accuracy: 0.9410530457019212
Confusion Matrix:
[[ 616   39    2    2    0    0  120  101]
 [    8 2058    0    0    0    7    7    5]
 [    0    0 2577    0    0   11    0    0]
 [    0    0    0 2773    0    0    0    0]
 [    0    0    0    0 2607    0    0    0]
 [    0    1    1    0    0 2403    0    0]
 [  130   26    5    1    0    7  411  187]
 [  102   12    0    0    0    7  118  907]]
Classification Report:
              precision    recall  f1-score   support

           0       0.72      0.70      0.71       880
           1       0.96      0.99      0.98      2085
           2       1.00      1.00      1.00      2588
           3       1.00      1.00      1.00      2773
           4       1.00      1.00      1.00      2607
           5       0.99      1.00      0.99      2405
           C       0.63      0.54      0.58       767
           X       0.76      0.79      0.77      1146

    accuracy                           0.94     15251
   macro avg       0.88      0.88      0.88     15251
weighted avg       0.94      0.94      0.94     15251
```

Figure 17: MLP

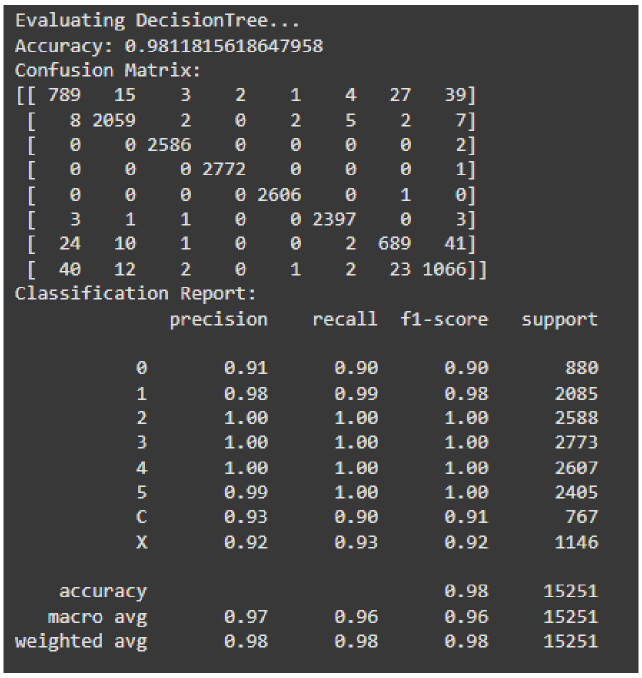
- Random Forest:



```
Evaluating RandomForest...
Accuracy: 0.9885253426004852
Confusion Matrix:
[[ 812   24    0    0    0    2   16   26]
 [    0 2085    0    0    0    0    0    0]
 [    0    0 2588    0    0    0    0    0]
 [    0    0    0 2773    0    0    0    0]
 [    0    0    0    0 2607    0    0    0]
 [    0    0    0    0    0 2405    0    0]
 [   21    7    0    0    0    0  699   40]
 [   18    4    0    0    0    0   17 1107]]
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.92      0.94       880
           1       0.98      1.00      0.99      2085
           2       1.00      1.00      1.00      2588
           3       1.00      1.00      1.00      2773
           4       1.00      1.00      1.00      2607
           5       1.00      1.00      1.00      2405
           C       0.95      0.91      0.93       767
           X       0.94      0.97      0.95      1146

    accuracy                           0.99     15251
   macro avg       0.98      0.98      0.98     15251
weighted avg       0.99      0.99      0.99     15251
```

Figure 18: Random Forest

- Decision Tree:



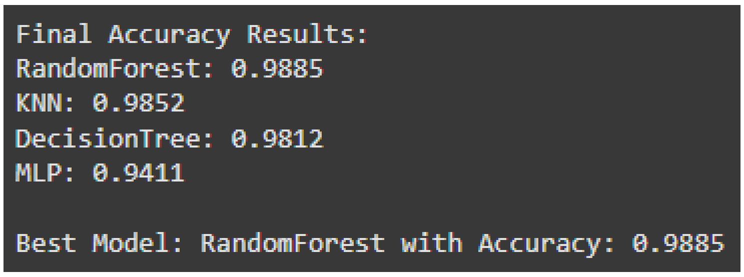
```
Evaluating DecisionTree...
Accuracy: 0.9811815618647958
Confusion Matrix:
[[ 789   15    3    2    1    4   27   39]
 [    8 2059    2    0    2    5    2    7]
 [    0    0 2586    0    0    0    0    2]
 [    0    0    0 2772    0    0    0    1]
 [    0    0    0    0 2606    0    1    0]
 [    3    1    1    0    0 2397    0    3]
 [   24   10    1    0    0    2  689   41]
 [   40   12    2    0    1    2   23 1066]]
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.90      0.90       880
           1       0.98      0.99      0.98      2085
           2       1.00      1.00      1.00      2588
           3       1.00      1.00      1.00      2773
           4       1.00      1.00      1.00      2607
           5       0.99      1.00      1.00      2405
           C       0.93      0.90      0.91       767
           X       0.92      0.93      0.92      1146

    accuracy                           0.98     15251
   macro avg       0.97      0.96      0.96     15251
weighted avg       0.98      0.98      0.98     15251
```

Figure 19: Decision Tree

8. CONCLUSION

Conclusion:

- We Found that the Random Forest Model was best at predicting credit card scores approval. Followed by, KNN and the rest of the models used.



```
Final Accuracy Results:
RandomForest: 0.9885
KNN: 0.9852
DecisionTree: 0.9812
MLP: 0.9411

Best Model: RandomForest with Accuracy: 0.9885
```

Figure 20: Models Comparison

- Following the steps that were given to us and understanding them and implementing them in python and seeing results helped us understand these concepts and allowed us to take a step forward in our learning path.