School of Business and Management

BUSM131 – Masterclass in Business Analytics

Credit Card Prediction

Devansh Verma

220560366

List of Contents

Abstract:

This project involves the development and evaluation of a machine learning model for predicting credit card defaults. The primary objective of this project is to help a bank reduce its credit card default rate and improve its risk management strategy. The model takes into account several features such as the customer's credit history, payment history, outstanding balances, and other factors that contribute to credit card defaults which has been provided

The model's potential applications are diverse and numerous, including developing targeted marketing strategies, offering personalized credit limits, improving risk management, and even identifying potentially fraudulent activities. The report also conducts feature engineering to encode the dataset for use in different machine learning models and techniques such as Random Forest, KNN and ANN.

Introduction:

Credit cards have become a ubiquitous financial instrument that allows consumers to make purchases and pay for them over time. The Banks provide credit cards to consumers as a way to earn interest on the balance carried by the cardholders. The bank here aims to reduce their credit card default rate and improve their risk management strategy. To achieve this, they want to identify the key factors that contribute to credit card defaults and develop a model that can predict the likelihood of a customer defaulting on their credit card payments. By identifying the key factors that contribute to credit card defaults, the bank can develop targeted strategies to reduce the default rate and improve their risk management.

The effectiveness of machine learning methods in credit card prediction systems, particularly in detecting fraud and predicting credit risk. They highlight the potential of machine learning to improve the accuracy and robustness of credit scoring systems, which can have significant implications for the financial industry.

In this study, the authors compared the four different machine learning algorithms for credit card fraud detection: decision tree, naive Bayes and artificial neural network. They use a random dataset of credit card transactions in which some of the data was falling under fraud categories.The authors observed that ANN was superior in terms of accuracy and sensitivity. (Awoyemi, 2017)

In another study it was observed that by using ML models including SVM, Naïve Bayes, and Random Forest algorithms. Performance criteria such as accuracy, precision, recall, and f1-score are analyzed using a confusion matrix. It was finally observed that Random Forest is found to have the best performance and is considered the optimal algorithm for detecting fraud. (IJRASET, 2022)

Credit risk is crucial for financial organisations to determine their strategies. However, increase in risks due to lack of lender experience and incomplete borrower credit history. To solve this, machine learning techniques are used to accurately predict credit risk. This study sets different credit risk scoring models using dataset from a real social lending platform. (Author links open overlay panelVincenzo Moscato et al., 2020)

Business Problem:

The bank has a goal to lower their credit card default rate and enhance their risk management strategy. To accomplish this objective, they intend to pinpoint the significant factors that lead to credit card defaults and construct a predictive model that can estimate the probability of a customer defaulting on their credit card payments. By detecting the essential factors that contribute to credit card defaults, the bank can create focused plans to lower the default rate and improve their risk management. In essence, they plan to utilize data analysis and machine learning techniques to examine the elements that heighten the probability of credit card defaults, and use this knowledge to make informed decisions when approving credit card applications and determining credit limits. This will help the bank decrease financial losses, enhance customer satisfaction, and maintain compliance with regulations.

In recent years, credit card defaults have become a growing concern for banks, and the need for effective risk management strategies has become more critical.

for instance, Mastercard: Mastercard, a global payment technology company, has developed a machine learning-powered fraud detection system that analyzes transaction data in real-time to detect and prevent fraudulent activity. The system uses advanced analytics and machine learning algorithms to identify patterns and anomalies in transaction data and flag potential fraud. According to Mastercard, the system has resulted in a 60% reduction in fraud losses. (PYMNTS.com, 2020)

Similarly, HSBC, a multinational banking and financial services company, has developed a machine learning-powered credit decisioning system that uses a wide range of data points, including credit score, income, employment status, and payment behavior, to evaluate credit risk and make more accurate credit decisions. The system is designed to improve the speed and accuracy of credit decisions while also minimizing the risk of defaults. (Donnelly, 2017)

The organizations that face and solve similar problems using business analytics and investing in machine learning:

American Express: American Express, a global financial services company, has developed a machine learning-powered fraud detection system that uses advanced analytics to identify fraudulent transactions in real-time. The system is trained on a vast dataset of past transactions and can detect unusual patterns of behavior that may indicate fraud. (*American Express, 2023)*

PayPal: PayPal, a leading digital payment platform, has developed a machine learning-powered credit risk management system that uses data analytics to evaluate creditworthiness and determine credit limits for customers. The system considers a wide range of data points, including credit score, payment history, and transaction data, to make informed credit decisions.(*Home, 2023)*

Capital One: Capital One, a large US bank, has developed a machine learning-powered credit decisioning system that uses advanced analytics to evaluate credit risk and make informed credit decisions. The system considers a wide range of data points, including credit history, employment status, and payment behavior, to determine credit limits and interest rates for customers.(Capital,2018)

DATA, EDA AND METHODS :

The Main data which we used was provided at -

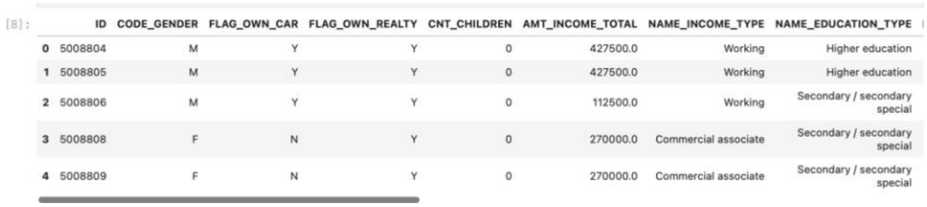
https://www.kaggle.com/datasets/rikdifos/credit-card-approval-prediction

The dataset underwent several cleaning and preprocessing steps to prepare it for analysis. These steps included removing duplicate records, checking for missing values, and correcting data types.

CREDIT CARD APPLICATION DATA			
S. NO	**COLUMN NAME**	**DATA TYPE**	**EXPLANATION**
1	ID	Numerical	Client's Number
2	CODE_GENDER	Categorical	Client's Gender
3	FLAG_OWN_CAR	Categorical	If the Client owns a car
4	FLAG_OWN_REALITY	Categorical	If the Client owns a property
5	CNT_CHILDREN	Numerical - Integer	Client's Number of Children
6	AMT_INCOME_TOTAL	Numerical - Float	Annual Income of the Client
7	NAME_INCOME_TYPE	Categorical	Income Category
8	NAME_EDUCATION_TYPE	Categorical	Client's Education Level
9	NAME_FAMILY_STATUS	Categorical	Marital Status of the Client
10	NAME_HOUSING_TYPE	Categorical	Housing type of the Client
11	DAYS_BIRTH	Numerical - Integer	Client's Birth date
12	DAYS_EMPLOYED	Numerical - Integer	Start date of employment
13	FLAG_MOBIL	Numerical - Integer	If the Client owns a Mobile Phone
14	FLAG_WORK_PHONE	Numerical - Integer	If the Client owns a Work Phone
15	FLAG_PHONE	Numerical - Integer	If the Client owns a Landline Phone

16	FLAG_EMAIL	Numerical - Integer	If the Client has an EMAIL ID
17	OCCUPATION_TYPE	Categorical	Client’s Occupation Type
18	CNT_FAM_MEMBERS	Numerical - Float	Number of members in the Client’s Family
19	STATUS	Categorical	Our Target Variable
20	DEFAULT	Categorical	

Table. 1 (DataSet)



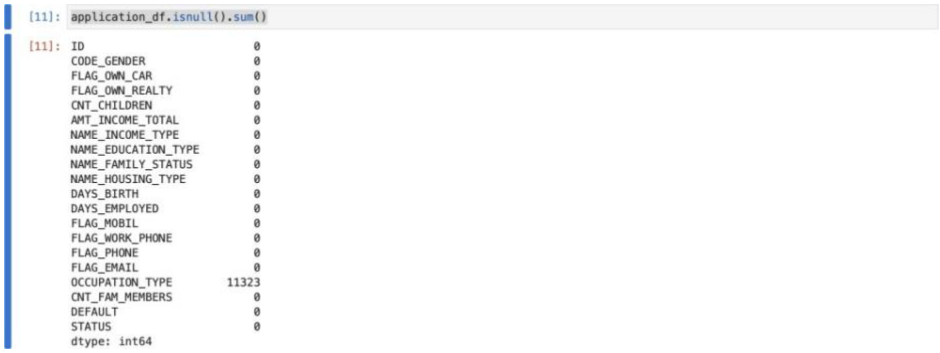
[8]:

	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

Fig. 1 (Orignal DataSet)

The Data Set Csv file was loaded into Jupyter Notebook (applications_with_status.csv). This was the raw data which contained 36457 rows and 20 columns. After this we looked for any missing values in the Data Set. This function is useful for identifying which cells in the dataframe contain missing values so that appropriate actions can be taken to handle them.

application_df.isnull().sum() then we ran this command to check the count of missing values in each column of the dataframe. The column named OCCUPATION_TYPE had the most amount of missing values.

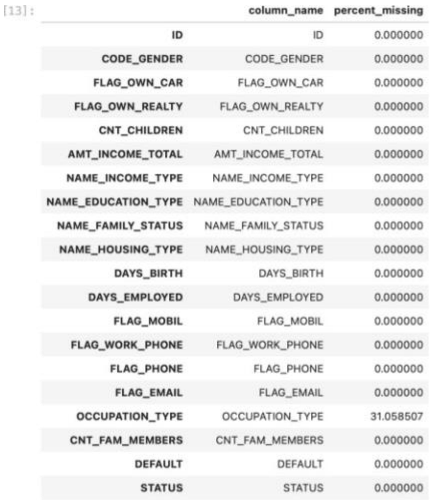


```
[11]: application_df.isnull().sum()
```

```
[11]: 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        11323
      CNT_FAM_MEMBERS            0
      DEFAULT                    0
      STATUS                     0
      dtype: int64
```

Fig. 2 (Count of Missing Values)

Then we converted these missing values to percentage which gives out 31% for the Column OCCUPATION_TYPE



[13]:

	column_name	percent_missing
ID	ID	0.000000
CODE_GENDER	CODE_GENDER	0.000000
FLAG_OWN_CAR	FLAG_OWN_CAR	0.000000
FLAG_OWN_REALTY	FLAG_OWN_REALTY	0.000000
CNT_CHILDREN	CNT_CHILDREN	0.000000
AMT_INCOME_TOTAL	AMT_INCOME_TOTAL	0.000000
NAME_INCOME_TYPE	NAME_INCOME_TYPE	0.000000
NAME_EDUCATION_TYPE	NAME_EDUCATION_TYPE	0.000000
NAME_FAMILY_STATUS	NAME_FAMILY_STATUS	0.000000
NAME_HOUSING_TYPE	NAME_HOUSING_TYPE	0.000000
DAYS_BIRTH	DAYS_BIRTH	0.000000
DAYS_EMPLOYED	DAYS_EMPLOYED	0.000000
FLAG_MOBIL	FLAG_MOBIL	0.000000
FLAG_WORK_PHONE	FLAG_WORK_PHONE	0.000000
FLAG_PHONE	FLAG_PHONE	0.000000
FLAG_EMAIL	FLAG_EMAIL	0.000000
OCCUPATION_TYPE	OCCUPATION_TYPE	31.058507
CNT_FAM_MEMBERS	CNT_FAM_MEMBERS	0.000000
DEFAULT	DEFAULT	0.000000
STATUS	STATUS	0.000000

Fig. 3 (Percentage of Missing Values)

We created a matrix plot which will visualizes the locations of missing values in the application_df dataframe.

Each row in the matrix represents a column and each column represents a row. Each cell in the matrix is colored on the basis of number of missing values in that location. If a cell is white, it means that there are no missing values. If a cell is shaded in gray, it means that there is at least one missing value.

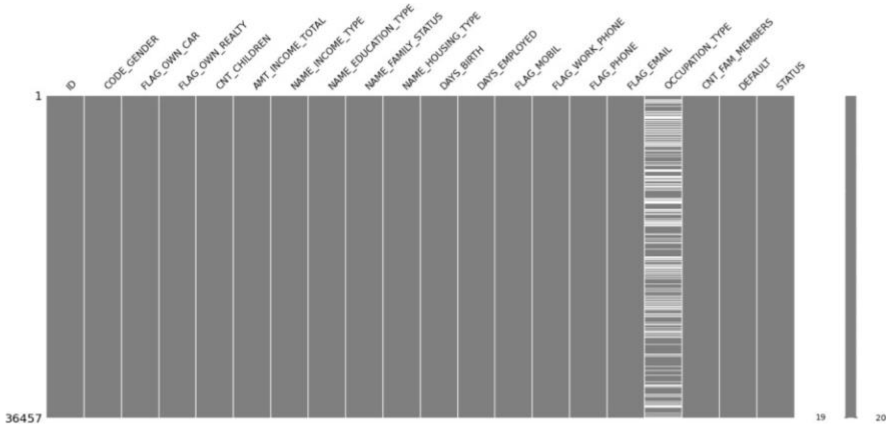


Fig. 4 (Matrix plot with missing values)

We dropped all the null values in the datafrafe resulting in this matrix

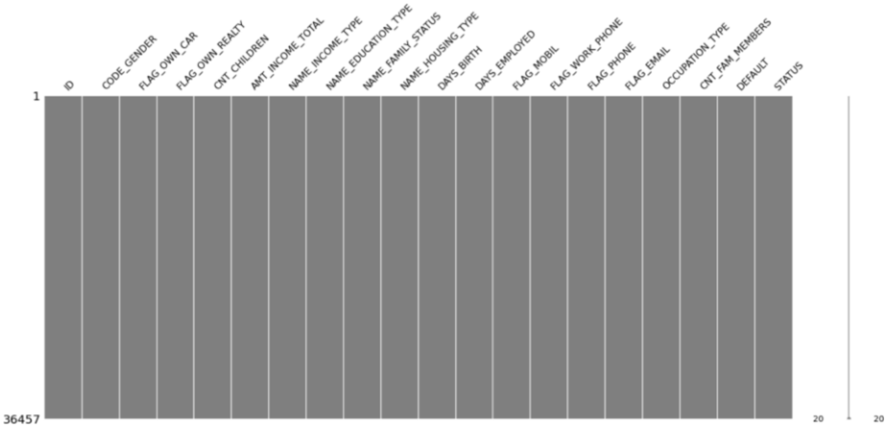


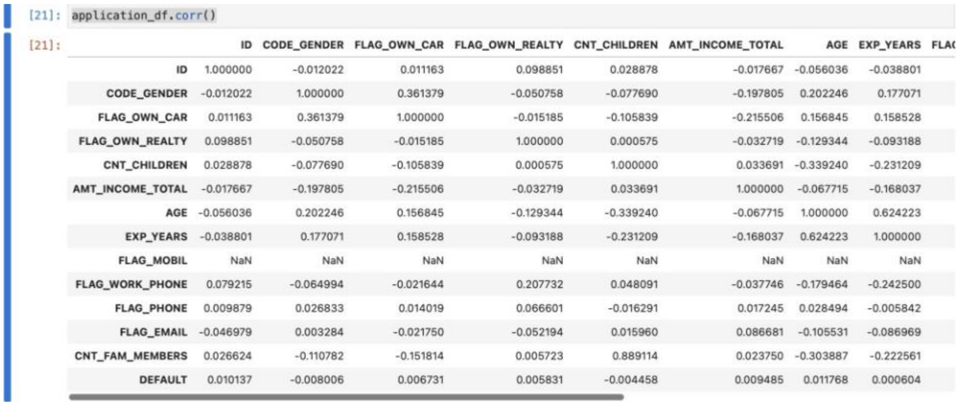
Fig. 5 (Matrix plot after removing null values)

We created a copy of our orignal data frame and converted the 'CODE_GENDER', 'FLAG_OWN_CAR', and 'FLAG_OWN_REALTY' columns into boolean values. In this case, the M value in the CODE_GENDER column is replaced with 0, the F value is replaced with 1, the Y value in the FLAG_OWN_CAR and FLAG_OWN_REALTY columns is replaced with 0, and the N value is replaced with 1. We also convert the 'DAYS_BIRTH' and 'DAYS_EMPLOYED' columns from the number of days to the number of years. We also replaced or renamed the 'DAYS_BIRTH' column to 'AGE', and the 'DAYS_EMPLOYED' column to 'EXP_YEARS'.

<u>EDA:</u>

EDA is a critical process that can help ensure that subsequent analysis or modeling tasks are based on accurate and reliable data, and can help uncover interesting patterns or insights that may not be immediately apparent.

Here we used the corr() function in pandas calculates the pairwise correlation between all columns in the application_df dataframe which results dataframe will have the same number of rows and columns as the original but with correlation coefficient ranges between -1 and 1, if the value is closer to 1 it indicates a stronger positive correlation between the two variables, values closer to –1 it indicates a stronger negative correlation, and values are closer to 0 it indicates the little to no correlation between the variables.



```
[21]: application_df.corr()
```

[21]:

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	AGE	EXP_YEARS	FLA
ID	1.000000	-0.012022	0.011163	0.098851	0.028878	-0.017667	-0.056036	-0.038801	
CODE_GENDER	-0.012022	1.000000	0.361379	-0.050758	-0.077690	-0.197805	0.202246	0.177071	
FLAG_OWN_CAR	0.011163	0.361379	1.000000	-0.015185	-0.105839	-0.215506	0.156845	0.158528	
FLAG_OWN_REALTY	0.098851	-0.050758	-0.015185	1.000000	0.000575	-0.032719	-0.129344	-0.093188	
CNT_CHILDREN	0.028878	-0.077690	-0.105839	0.000575	1.000000	0.033691	-0.339240	-0.231209	
AMT_INCOME_TOTAL	-0.017667	-0.197805	-0.215506	-0.032719	0.033691	1.000000	-0.067715	-0.168037	
AGE	-0.056036	0.202246	0.156845	-0.129344	-0.339240	-0.067715	1.000000	0.624223	
EXP_YEARS	-0.038801	0.177071	0.158528	-0.093188	-0.231209	-0.168037	0.624223	1.000000	
FLAG_MOBIL	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
FLAG_WORK_PHONE	0.079215	-0.064994	-0.021644	0.207732	0.048091	-0.037746	-0.179464	-0.242500	
FLAG_PHONE	0.009879	0.026833	0.014019	0.066601	-0.016291	0.017245	0.028494	-0.005842	
FLAG_EMAIL	-0.046979	0.003284	-0.021750	-0.052194	0.015960	0.086681	-0.105531	-0.086969	
CNT_FAM_MEMBERS	0.026624	-0.110782	-0.151814	0.005723	0.889114	0.023750	-0.303887	-0.222561	
DEFAULT	0.010137	-0.008006	0.006731	0.005831	-0.004458	0.009485	0.011768	0.000604	

Fig.6 (Pairwise Correlation Dataset)

After this we dropped the columns 'ID', 'CNT_CHILDREN', 'FLAG_MOBIL', 'FLAG_WORK_PHONE', 'FLAG_EMAIL', 'DEFAULT', 'FLAG_OWN_CAR', 'CODE_GENDER', and 'FLAG_PHONE' using the columns parameter. The inplace=True parameter is used to modify the application_df dataframe directly with the axis = 1.

We then converted some columns to ‘bool’, ‘string’ etc. Which has been mentioned in the comments.

The uncommented code converts the 'STATUS' column in the application_df dataframe to an int64 data type using the astype() function in pandas.

After this we ran a command **application_df.corrwith(application_df['STATUS'])*100** this executed/ computes the correlation between each column of the application_df dataframe and the 'STATUS' column, which is multiplied by 100 to make it as a percentage.

This is very useful for identifying which variables are more strongly associated with the 'STATUS' variable. Variables with the high positive correlations with the 'STATUS' variable are considered as predictors of default, while variables with high negative correlations with the 'STATUS' variable are considered as potential protective factors against default.

After this we created list called **num_cols** which contained the names of all columns in the application_df dataframe where the data type is not 'O’. It selects only those columns where the data type is not equal to 'O', we also create another list called **str_cols** that contains the names of all columns in the application_df dataframe where the data type is an object 'O’. It selects only those columns where the data type is equal to 'O'.

Correlation Matrix (bivariate analysis)

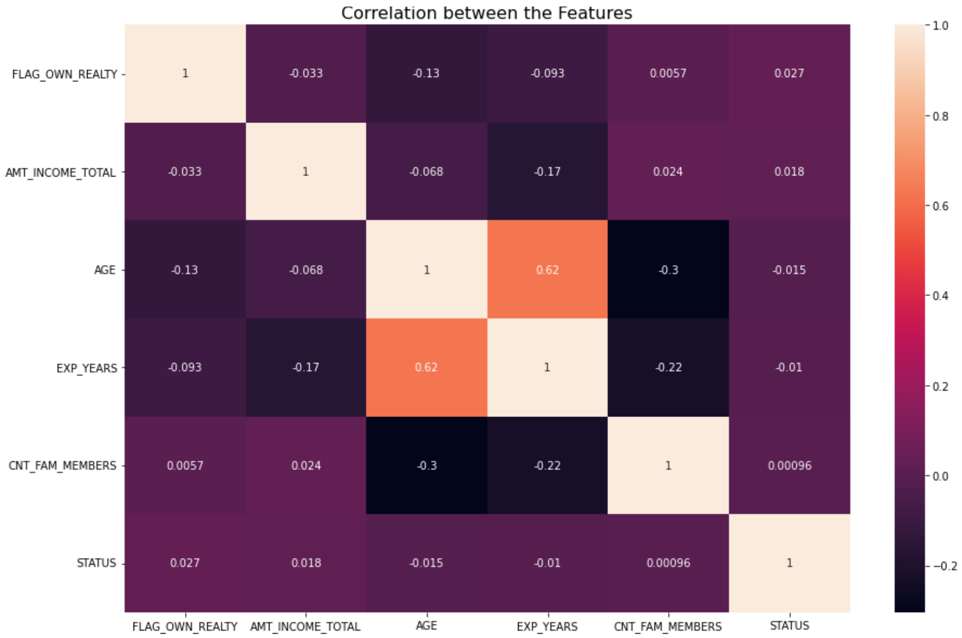


Fig. 8 (Correlation Matrix)

The heatmap or the matrix visualizes the pairwise correlations between all pairs of variables in the application_df dataframe, The darker colors indicates the much more stronger positive or negative correlations, and the lighter colors indicates weaker or no correlations at all.

This matrix is between different columns in the datafeame - 'FLAG_OWN_REALTY', 'AMT_INCOME_TOTAL', 'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE', 'NAME_FAMILY_STATUS', 'NAME_HOUSING_TYPE', 'AGE','EXP_YEARS', 'OCCUPATION_TYPE', 'CNT_FAM_MEMBERS', 'STATUS'. Also, all these columns have the same data type (dtype) which is object.

Chi-Square Test:

The chi-square test is a statistical method which is used to find if there is a significant association between two categorical variables. It also compares the observed frequencies of each category in a table to the expected frequencies, keeping in mind that there is no association between the variables. The p-value is calculated in a chi-square test, which indicates strong evidence against the null hypothesis and that there is a significant association between the two variables when the p-value is small, On the other hand, a large p-value suggests visa-versa.

The chi-square test which we performed was on each variable in the variables list to find whether there is a significant relation between that variable and the loan status which we have mentioned by the "STATUS" column.

The chi2_contingency() function from the scipy.stats library is then used to perform the chi-square test which results in chi-square statistic, degrees of freedom, and p-value.

We have created a histograms for each of the numerical columns that are in the application_df DataFrame and we have done this by using Seaborn's histplot function. We also made pair plot using the Seaborn library. The plot which we made includes scatter plots for each pairwise combination of the numerical variables and histograms for each individual variable. We made the ‘STATUS’ as the hue parameter in this case.

The Results:

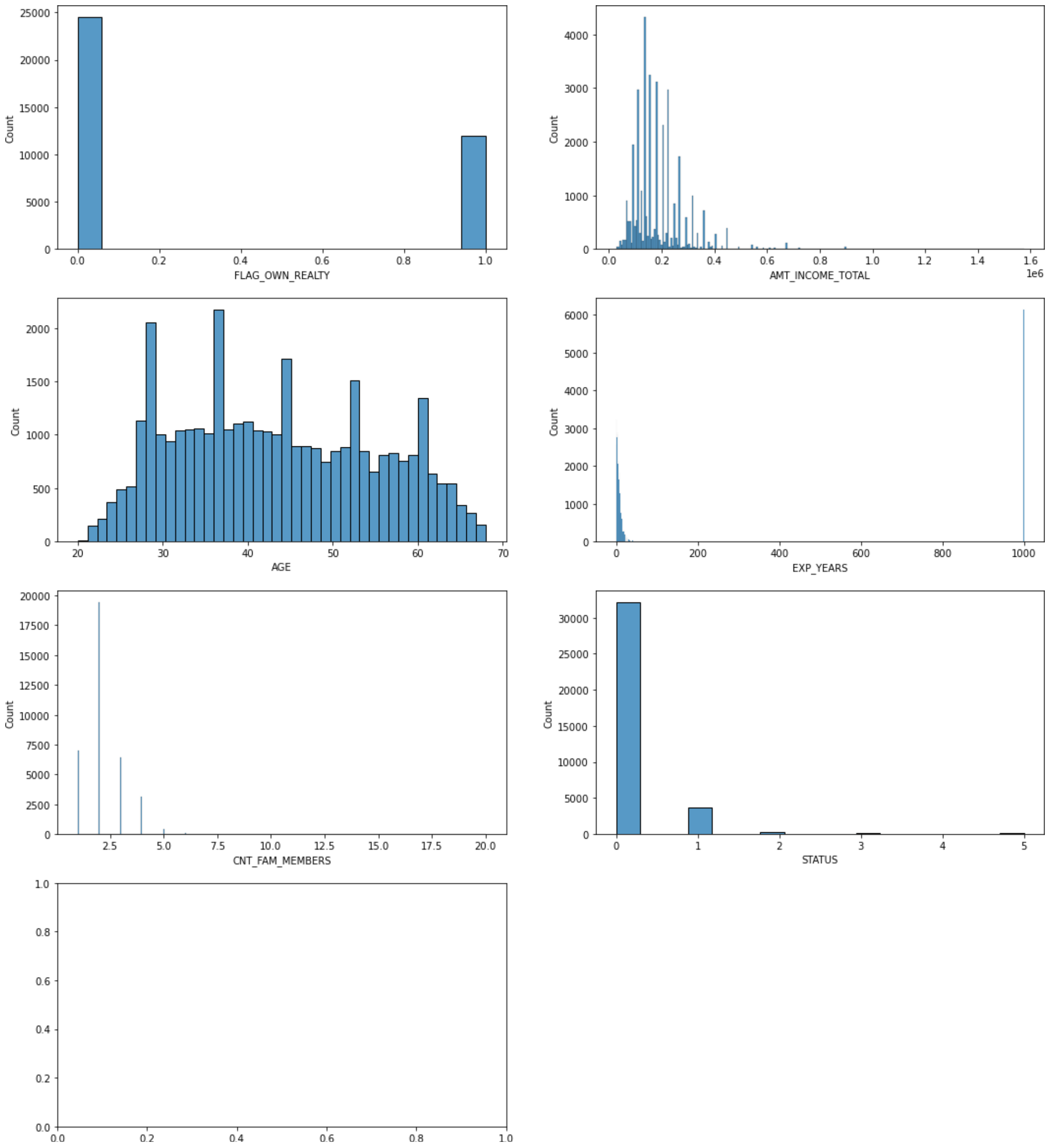


Fig.7 (Histographs)

Each histogram represents different columns of the main data frame, which represents the distribution of values for each numerical variable in the dataset.

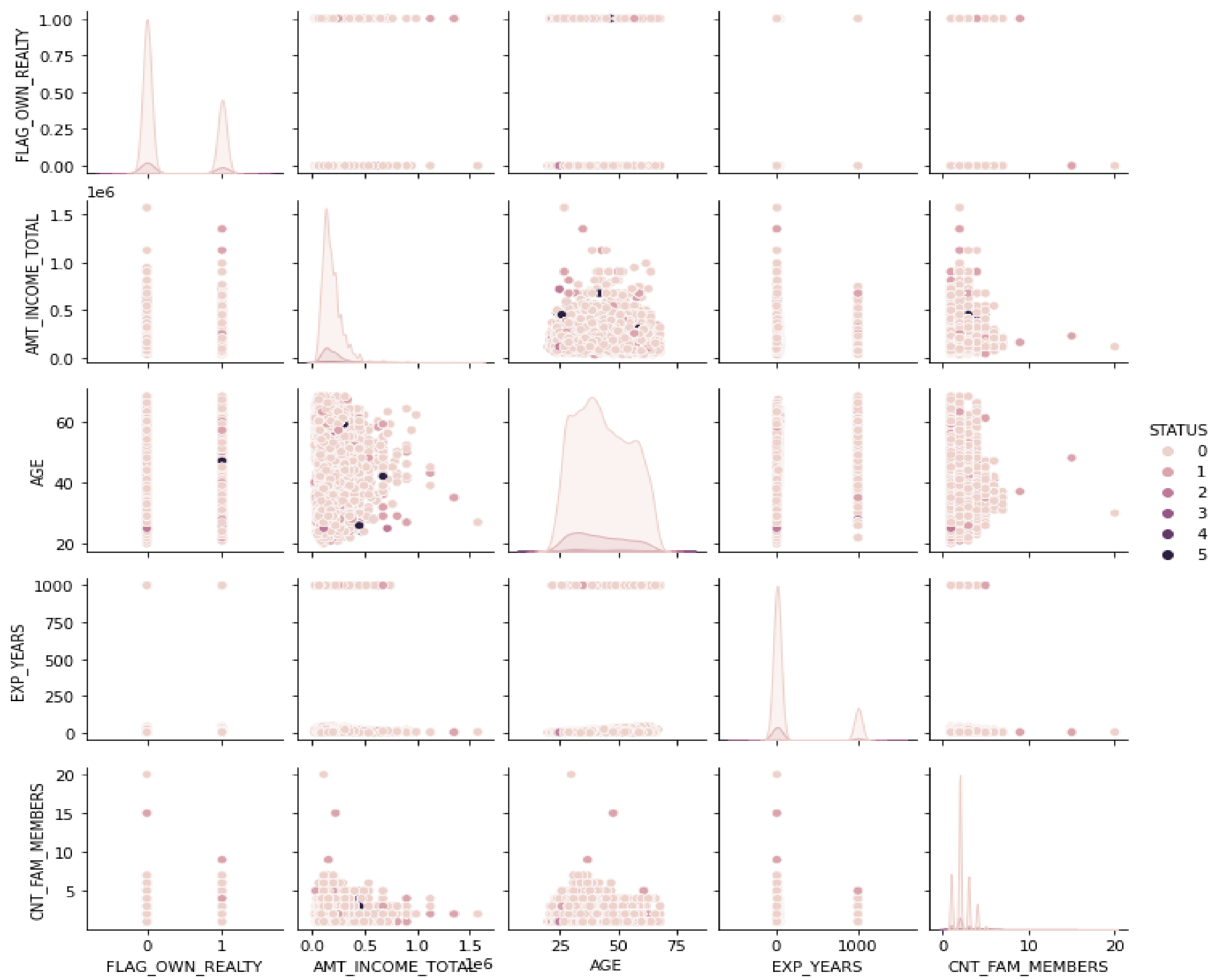


Fig.8 (Scatter plot)

<u>Pie Chart:</u>

We then created a pie-chart to show different types of education. We first counted the frequency of each education type using the value_counts() and calculated the proportion of each type using the total count of education types.

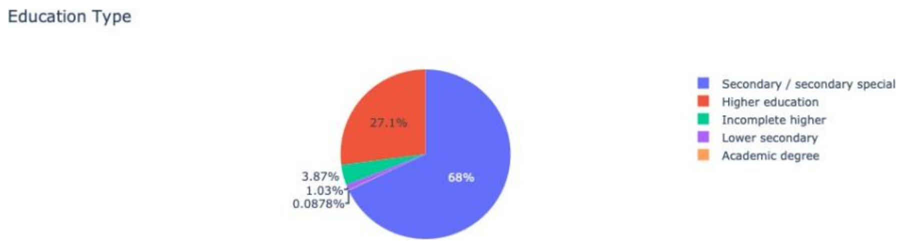
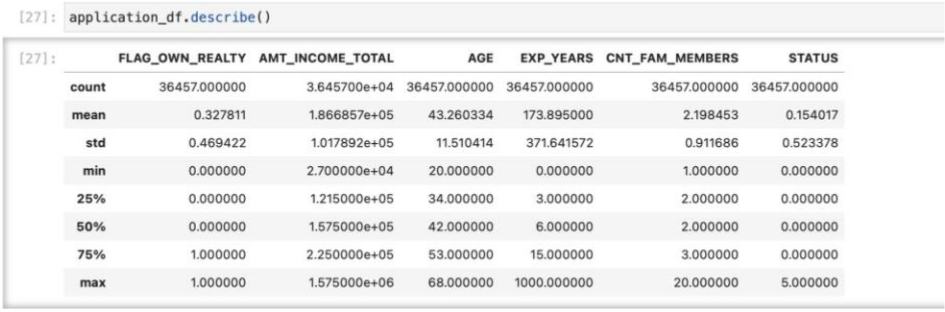


Fig.9 (Pie Chart)

We provided the viewers with DataPrep Report to make them understandthe data easily. It includes DataSet Statistics, DataSet Insights. We have also provided it for the all 20 columns aswell.

We have provided univariant statistics aswell:



```
[27]: application_df.describe()
```

[27]:

	FLAG_OWN_REALTY	AMT_INCOME_TOTAL	AGE	EXP_YEARS	CNT_FAM_MEMBERS	STATUS
count	36457.000000	3.645700e+04	36457.000000	36457.000000	36457.000000	36457.000000
mean	0.327811	1.866857e+05	43.260334	173.895000	2.198453	0.154017
std	0.469422	1.017892e+05	11.510414	371.641572	0.911686	0.523378
min	0.000000	2.700000e+04	20.000000	0.000000	1.000000	0.000000
25%	0.000000	1.215000e+05	34.000000	3.000000	2.000000	0.000000
50%	0.000000	1.575000e+05	42.000000	6.000000	2.000000	0.000000
75%	1.000000	2.250000e+05	53.000000	15.000000	3.000000	0.000000
max	1.000000	1.575000e+06	68.000000	1000.000000	20.000000	5.000000

Fig.10 (Statistics)

We have made a BarPlot between our Target Variable STATUS v/s OCCUPATION_TYPE

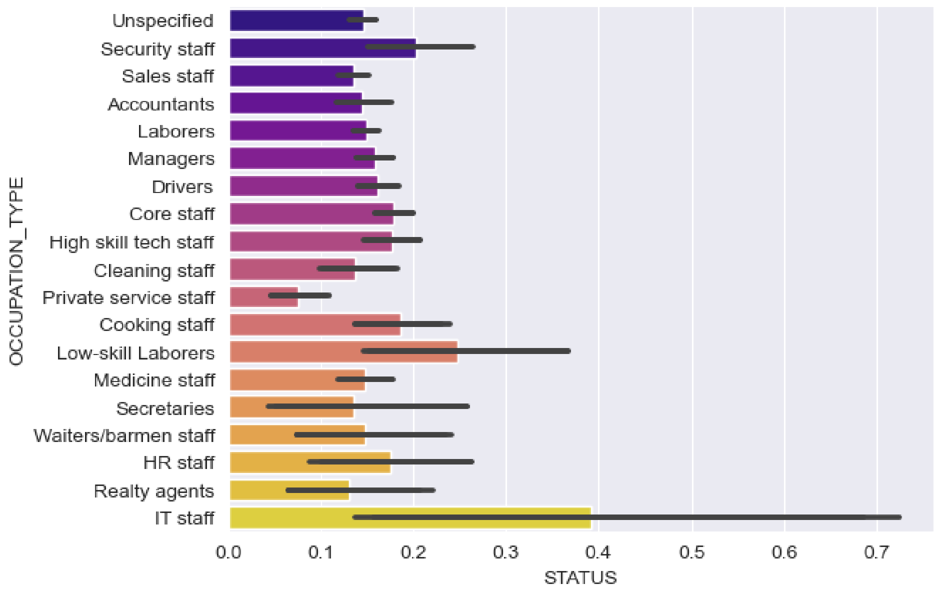


Fig.11 (Bar Plot)

We have made count plot aswell, two defaults TRUE and FALSE, between COUNT and OCCUPATION_TYPE. This can help visualize any patterns or differences in the distribution of occupations between those who defaulted and those who did not.

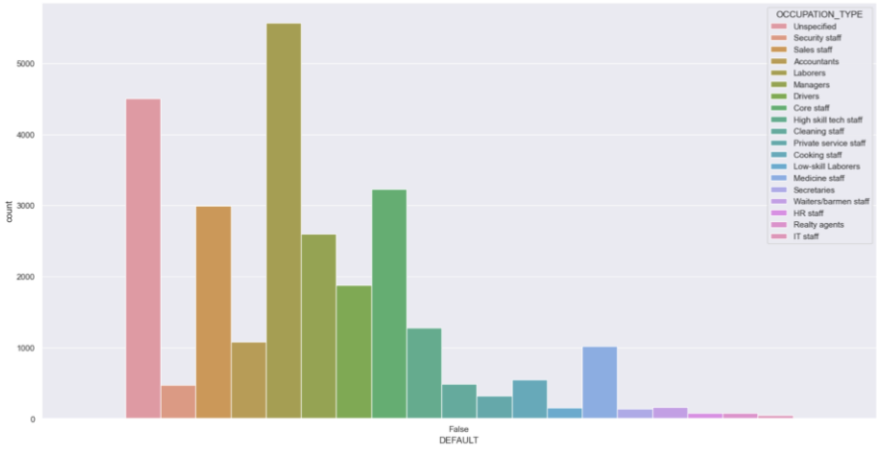


Fig.12 (Count vs False)

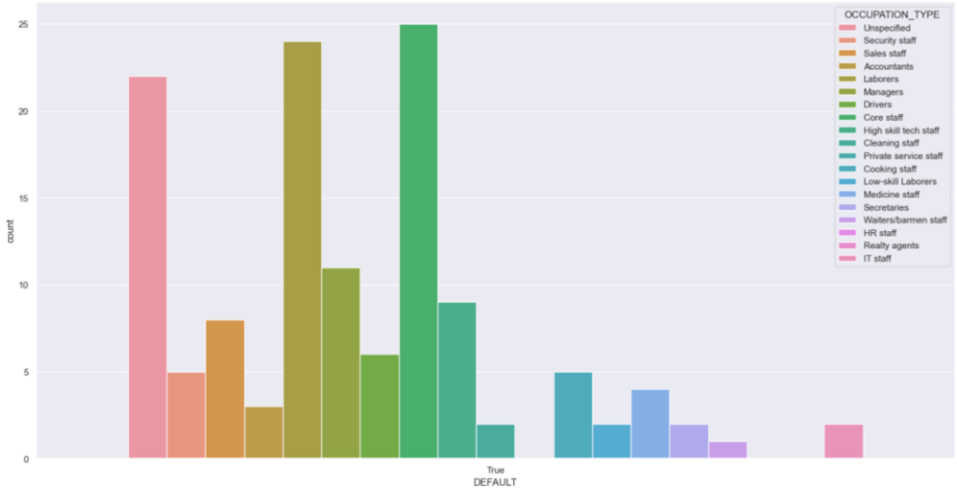


Fig.13 (Count vs True)

We did a test, train and validation split in our data. For this we did:

We ran a code which splits the dataset into three subsets:

- Training
- Validation
- Testing sets

We called up the function train_test_split() from scikit-learn and we specified the parameter test_size=0.2, which specifies that 20% of the data will be used for testing, while the remaining 80% will be used for training and validation. After this we again called up train_test_split() to split the remaining 80% training and validation data into two sets with a 75/25 ratio. By doing this it'll creates a final split of 60% training data, 20% validation data, and 20% testing data. We have set the random_state parameter 42 to make sure reproducibility of the results. By splitting the data into three separate sets, it is possible to develop and evaluate machine learning models that can accurately generalize to new data.

We also ran a command to split the feature variables into two groups: numerical and categorical. Since STATUS is the target variable and not a feature, it is removed from the list of categorical features. This allows us to apply different preprocessing techniques (such as normalization for numerical features and one-hot encoding for categorical features) to each group separately.

The Categorical features can further be divided in Ordinal and OneHotEncoded features. We discovered that the only feature suitable for an ordinal encoder is 'education'.

One-Hot-Encoder replaces categorical features by boolean features, stating whether a certain category is true or not!!

We run the pipeline to check, whether it runs with no problems

temp = one_hot_pipeline.fit_transform(x)

We further created a pipeline for preprocessing numerical features in a dataset with The 'StandardScaler' function standardizes the selected numerical features by subtracting the mean and dividing by the standard deviation.

At last we created a full pipeline for preprocessing a dataset that contains numerical, one-hot encoded, and ordinal features.

The ML Models we have used:

- Random Forest - It is a frequently employed machine learning algorithm that is suitable for both regression and classification tasks. It is an ensemble learning technique that combines many decision trees to produce a prediction. Random Forest is valued for its strong performance, ability to avoid overfitting, and capacity to handle datasets with many input features. It is also effective at learning from imbalanced datasets, making it a popular choice in these scenarios.
- KNN - K-Nearest Neighbors (KNN) is a widely-used machine learning algorithm that can be used for regression and classification problems. It is an instance-based learning technique that relies on a labeled database to make predictions for new, unlabeled data points. KNN is favored for its simplicity, versatility, and capacity to handle nonlinear data. It is suitable for both binary and multi-class classification problems, as well as regression problems. Additionally, KNN can accommodate both continuous and categorical input features.
- ANN - Artificial Neural Networks (ANNs) are a type of machine learning algorithm that can be applied to classification, regression, and clustering tasks. They are designed to emulate the structure and function of biological neurons in the brain, and can be used to model intricate and nonlinear relationships between input and output variables. ANNs are a versatile tool for a broad range of applications, natural language processing, and time-series analysis.

We used different Python Libraries like:

- Numpy
- Pandas
- Sklearn
- Keras
- Matplotlib
- Imblearn

Analysis and Results:

The table shows the chi-square statistic, degrees of freedom, and p-value for each variable. The chi-square statistic measures the difference between the observed and expected frequencies of the variable in the contingency table.

Variable	Chi-square statistic	Degrees of freedom	p-value

FLAG_OWN_REALTY	38.346836695541626	5	3.213787377269442e-07
AMT_INCOME_TOTAL	2562.898660575034	1320	3.219044188068273e-82
NAME_INCOME_TYPE	79.18948754900813	20	5.386892848695465e-09
NAME_EDUCATION_TYPE	62.024324230504064	20	3.446100928540676e-06
NAME_FAMILY_STATUS	107.6359528314522	20	5.288861194414323e-14
NAME_HOUSING_TYPE	49.168818105569216	25	0.002694514926981239
EXP_YEARS	297.6209401333129	220	0.00037806153772468613
OCCUPATION_TYPE	177.97235654787727	90	9.704827413497206e-08
CNT_FAM_MEMBERS	85.22137808677816	45	0.0002750357678367596

Table 2. (Results of Chi-square Test)

Starting with the ML Models we have used:

Random Forest:

We Create an instance of the Random Forest classifier with hyperparameters

We have set the n_estimators parameter which specifies the number of decision trees that will be constructed in the forest to 200, along with the max_depth parameter which specifies the maximum depth of each decision tree to 50 levels away from the root node and Finally, the random_state parameter which sets a seed value for the random number generator used to initialize the forest, ensuring that the same results will be produced each time the code is run with the same data.

Then we fit this model to the training data by By calling rf_clf.fit(prepared_smote,label_smote), the Random Forest classifier is trained on the already preprocessed dataset whith the use of the SMOTE technique which addresses class imbalance. Once the training is complete, the trained classifier can be used to make predictions on new, unseen data.

At last we Evaluated the model on the validation set which provides the accuracy score and tells us the proportion of correct predictions made by the classifier on the validation set along with the confusion matrix between actual and predicted values.

We got the accuracy to be 0.8024276377217554.

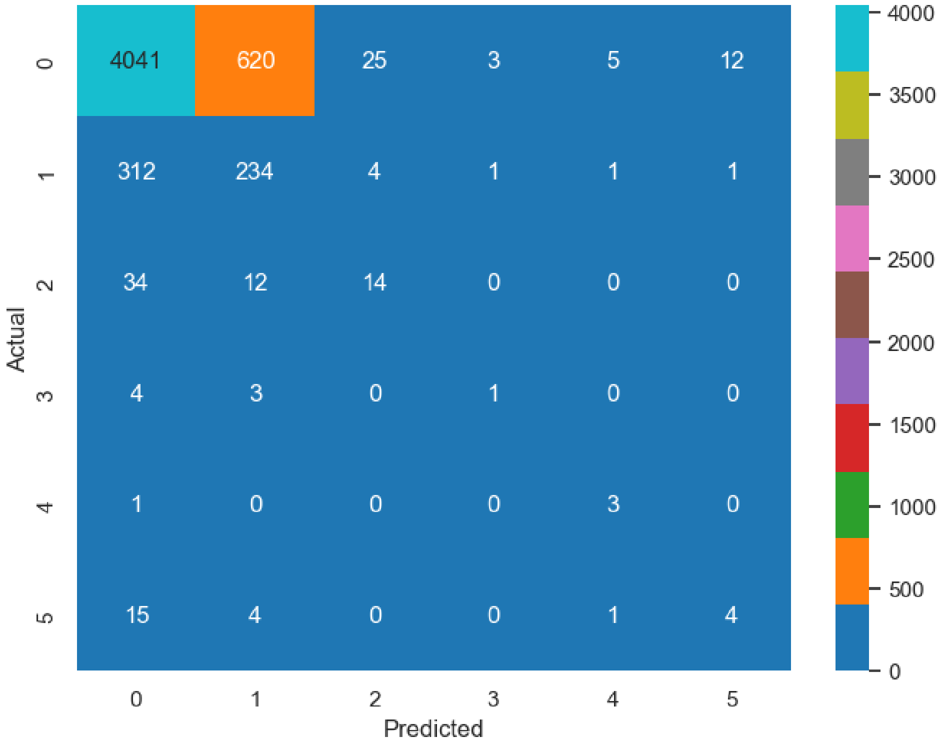


Fig.14 (RF Confusion Matrix)

KNN:

We created an instance of the KNeighborsClassifier with specifying the k=5, along with setting n_neighbors=5,weights='distance',p=1.

After this we fit this model to the training data againg by calling knn_clf.fit(prepared_smote,label_smote)

We have disccused the role of calling SMOTE functi0n in the above model.

At last we evaluated the model on the validation set which is used to predict the target labels for a validation set consisting of input features (df_prepared_val), and the predicted labels are assigned to a variable y_val_pred. The accuracy score provides a measure of the proportion of correct predictions made by the classifier on the validation set, along with the confusion matrix between actual and predicted values.

We got the accuracy to be 0.822782446311858

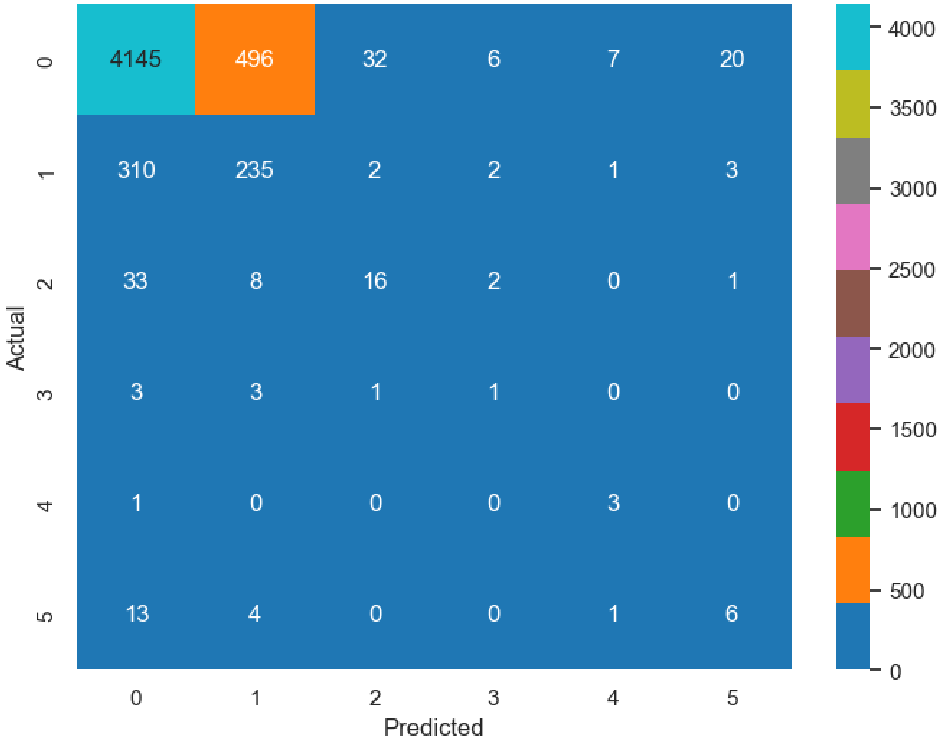


Fig. 14 (KNN Confusion Matrix)

ANN:

We created an instance of the MLPClassifier with hyperparameters with specifying hidden_layer_sizes=(100,), activation='relu', alpha=0.001, max_iter=500, random_state=42.

We then fit this model to the training data

mlp_clf.fit(prepared_smote,label_smote)

We have disccused the role of calling SMOTE functi0n in the above model.

We then finally evaluated the model on the validation set in which the mlp_clf.score() function provides the accuracy of the MLPClassifier on the validation set, which is the proportion of correct predictions made by the classifier.

We also made a confusion matrix that shows the number of true positives, true negatives, false positives, and false negatives.

We got the accuracy to be 0.6985994397759103

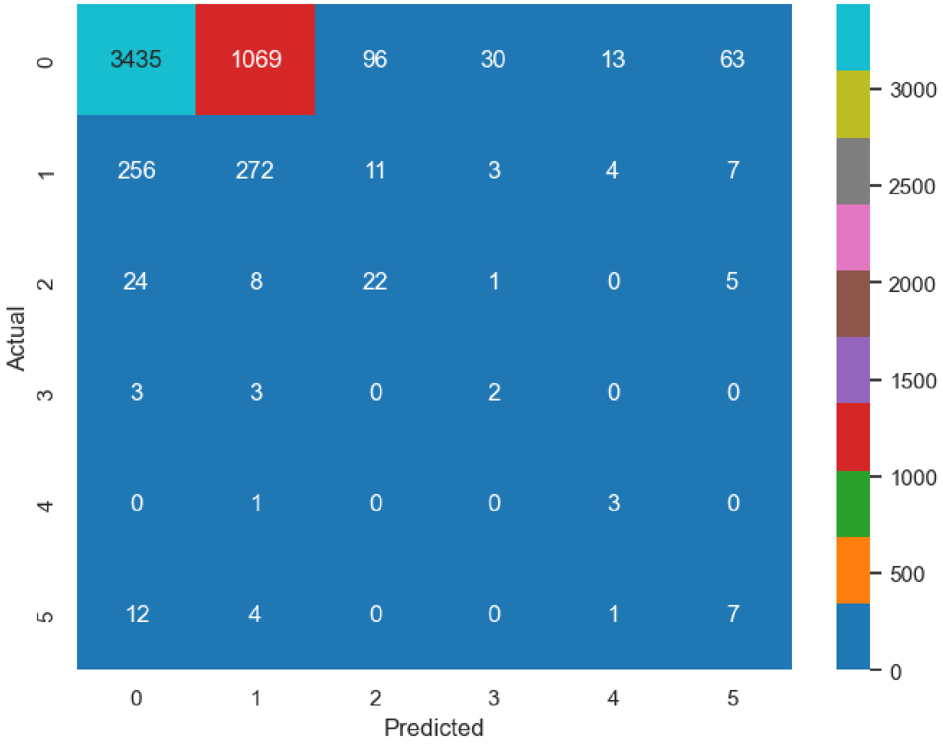


Fig. 15 (ANN Confusion Matrix)

The table below shows our result of the modelling with precision, recall, f1-score, support, accuracy, macro average and weighted average.

Random Forest	**precision**	**recall**	**f1-score**	**support**
0	0.92	0.86	0.89	4706
1	0.27	0.42	0.33	553
2	0.33	0.23	0.27	60
3	0.20	0.12	0.15	8
4	0.30	0.75	0.43	4
5	0.24	0.17	0.20	24
accuracy			0.80	5355
macro avg	0.37	0.43	0.38	5355
weighted avg	0.84	0.80	0.82	5355
KNN	**precision**	**recall**	**f1-score**	**support**

0	0.92	0.88	0.90	4706
1	0.32	0.42	0.36	553
2	0.31	0.27	0.29	60
3	0.09	0.12	0.11	8
4	0.25	0.75	0.38	4
5	0.20	0.25	0.22	24
accuracy			0.82	5355
macro avg	0.35	0.45	0.38	5355
weighted avg	0.85	0.82	0.83	5355
ANN	**precision**	**recall**	**f1-score**	**support**
0	0.92	0.73	0.81	4706
1	0.20	0.49	0.28	553
2	0.17	0.37	0.23	60
3	0.06	0.25	0.09	8
4	0.14	0.75	0.24	4
5	0.09	0.29	0.13	24
accuracy			0.70	

Table 3. (Result Comparision)

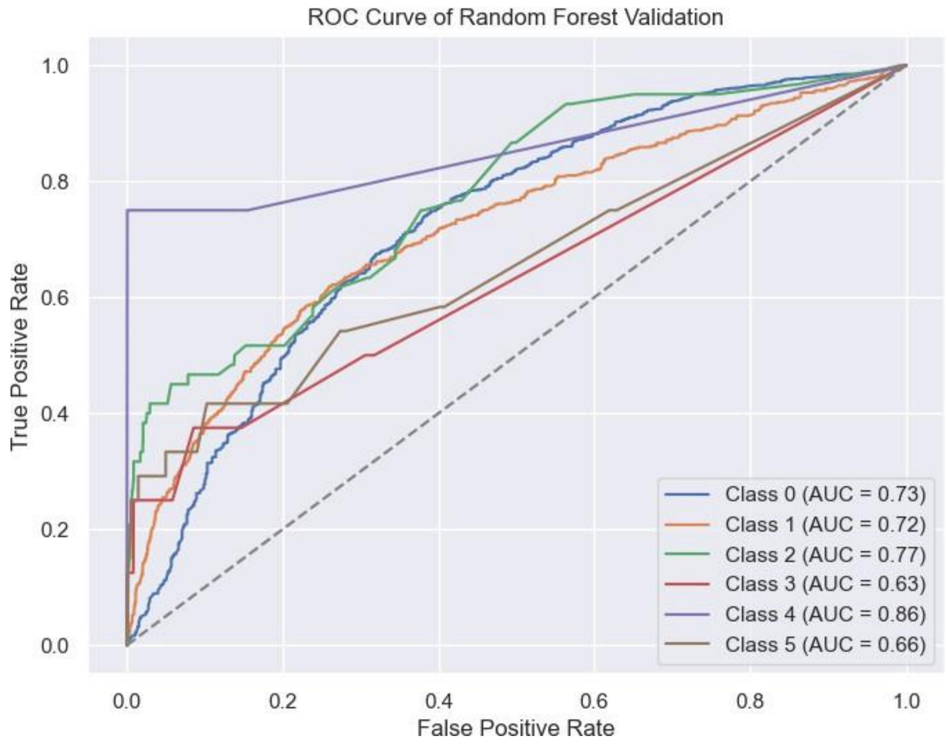
The Results for the ROC Curves for each Model:

The AUC (Area Under the Curve) is the scoring metric which is used for the evaluation of the performance of a classification model. The AUC score measures the area under the ROC curve, which plots the true positive rate (TPR) against the false positive rate (FPR) for different thresholds of the model's predicted probability.

Random Forest -

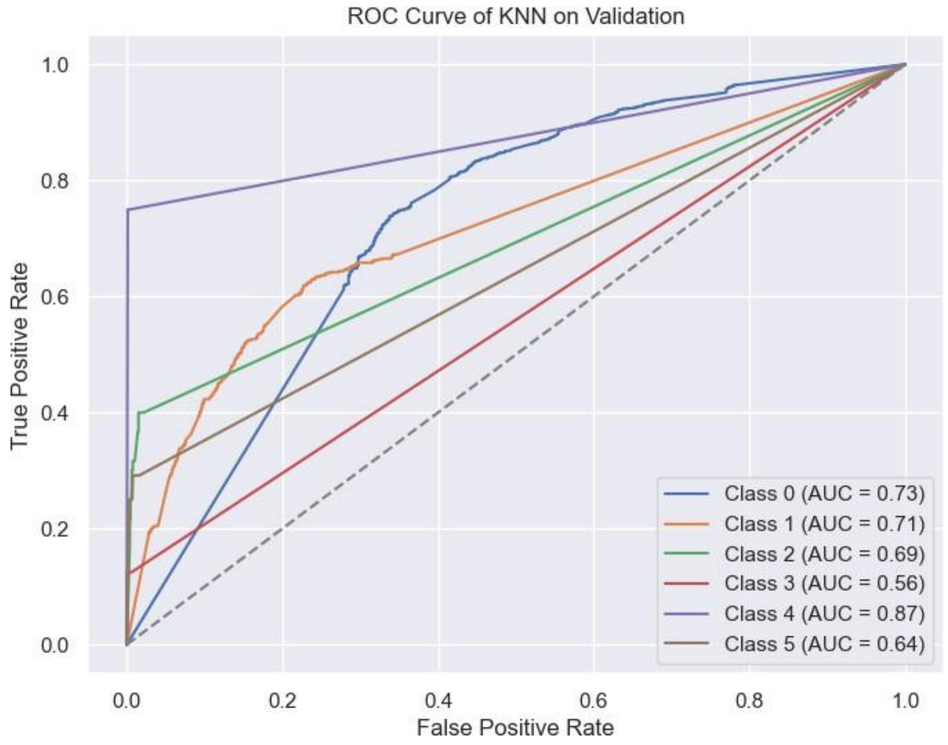
We plotted the ROC curve to get the predicted probability of each class which is from 0 to 5.

Also we computed the ROC curve and ROC AUC score for each class.



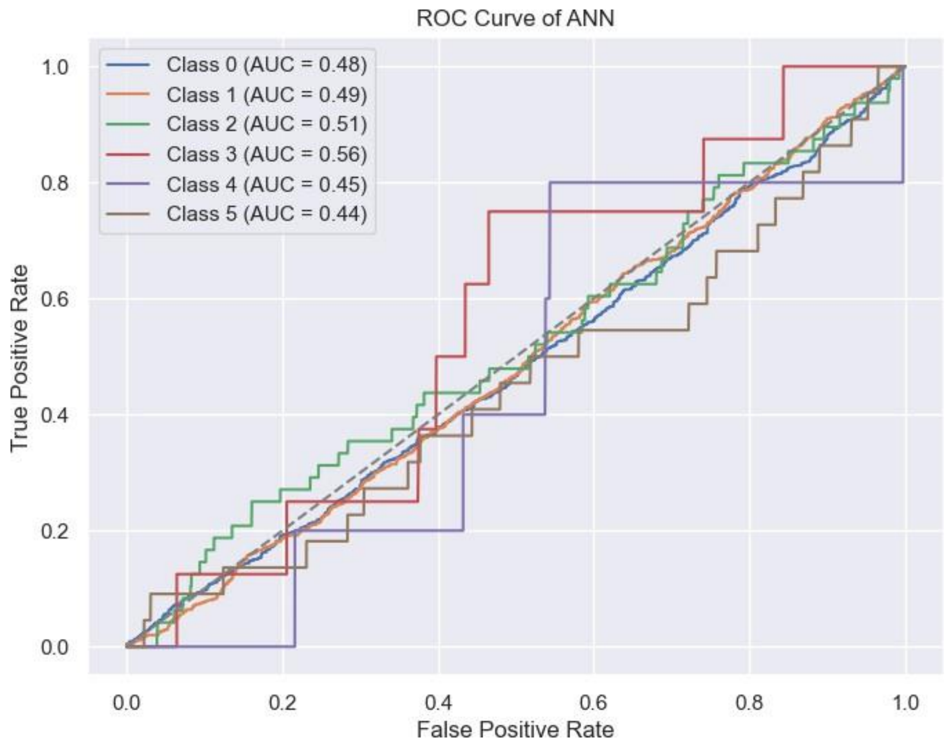
KNN -

Here also we plotted the ROC Curve for the each class.



ANN -

The ROC Curve for the ANN Model is shown below



Comparing the AUC scores for all the classes in the ROC Curve for all the Models.

Class	Random Forest (AUC)	KNN (AUC)	ANN (AUC)
0	0.73	0.73	0.48
1	0.72	0.71	0.49
2	0.77	0.69	0.51
3	0.63	0.56	0.56
4	0.86	0.87	0.45
5	0.66	0.64	0.44

Table 4 (Result Comparision ROC)

After testing and analysing the results the Random Forest is a better model choice than KNN and ANN for credit card prediction for several reasons:

Accuracy: It provides accurate predictions, which is important with the business problem where the model needs to identify whether a customer is likely to default on their credit card payment or not considering the risk factor as well. In contrast, KNN and ANN may not be as accurate as Random Forest.

Robustness to noise: The Credit card data which a bank may contain missing or corrupted values, which Random Forest can handle better than KNN and ANN due to its robustness to noisy data.

Reduced overfitting: It is better at handling overfitting than KNN and ANN by randomly selecting features for each tree and aggregating their results.

Interpretability: It provides main scores for each feature and makes it easier to interpret which variables are most important in predicting the target variable. This is very important in our business problem, where understanding which variables contribute most to the risk of default can be crucial.

Speed: Random Forest is comparatively faster compared to KNN and ANN, which may be important in cases where the prediction needs to be made quickly.

Business use case:

Assigning a risk scale based on status is very common in many industries, including finance and insurance. The aim to assign the risk scale is to classify customers into different risk categories based on their status.

We have assigned the risk scale 0-1-2 to those customers have lower risk. This means that they are less likely to default on payments or commit fraud and have less risk factors.

The moderate risk category is assigned for 3-4, which includes customers who are considered to have a higher risk than those in the low-risk category.

Finally, the high-risk category falls under 5 and includes customers who are considered to have the highest risk of default or fraud.

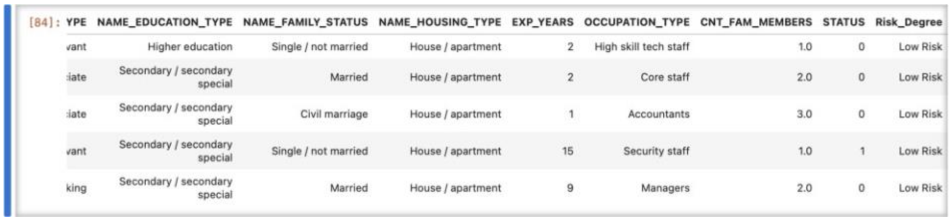
Assigning these risk scales based on status can benefit organizations manage their risk management.

0-1-2 = Low risk

3-4 = Moderate risk

5 = High risk

Further, we made a new column called 'Risk_Degree' which is based on the values in the 'STATUS' column. Where values 0, 1, and 2 are considered low risk, values 3 and 4 are considered moderate risk, and value 5 is considered high risk this finally adds the resulting values to a new column called 'Risk_Degree'.



[84]: YPE	NAME_EDUCATION_TYPE	NAME_FAMILY_STATUS	NAME_HOUSING_TYPE	EXP_YEARS	OCCUPATION_TYPE	CNT_FAM_MEMBERS	STATUS	Risk_Degree
vant	Higher education	Single / not married	House / apartment	2	High skill tech staff	1.0	0	Low Risk
:iate	Secondary / secondary special	Married	House / apartment	2	Core staff	2.0	0	Low Risk
:iate	Secondary / secondary special	Civil marriage	House / apartment	1	Accountants	3.0	0	Low Risk
vant	Secondary / secondary special	Single / not married	House / apartment	15	Security staff	1.0	1	Low Risk
king	Secondary / secondary special	Married	House / apartment	9	Managers	2.0	0	Low Risk

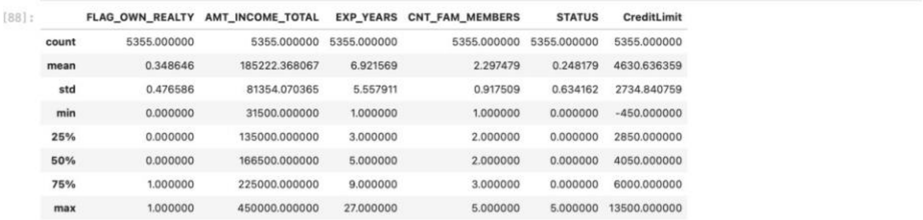
Fig.16 (Risk Degree)

Calculation of Credit Limit:

A new column is created which is named as Credit Limit. It uses debt-to-income (DTI) ratio as inputs and calculates a credit limit for each customer based on their income and debt. The function assumes a fixed monthly debt obligation of $1500.

The DTI is set to 0.4, which means that the customers can afford to have a maximum monthly debt equal to 40% of their monthly income.

After this the STATUS is set to 5, By updating the 'CreditLimit' column in this way, it can be used to reflect the updated credit limits for each customer based on their risk level.

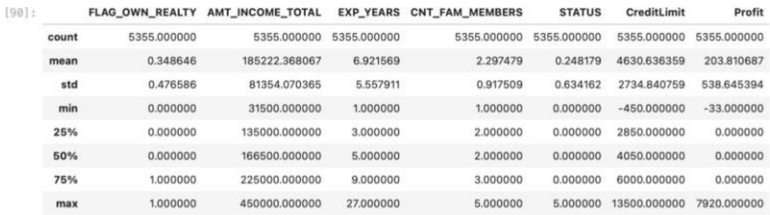


[88]:	FLAG_OWN_REALTY	AMT_INCOME_TOTAL	EXP_YEARS	CNT_FAM_MEMBERS	STATUS	CreditLimit
count	5355.000000	5355.000000	5355.000000	5355.000000	5355.000000	5355.000000
mean	0.348646	185222.368067	6.921569	2.297479	0.248179	4630.636359
std	0.476586	81354.070365	5.557911	0.917509	0.634162	2734.840759
min	0.000000	31500.000000	1.000000	1.000000	0.000000	-450.000000
25%	0.000000	135000.000000	3.000000	2.000000	0.000000	2850.000000
50%	0.000000	166500.000000	5.000000	2.000000	0.000000	4050.000000
75%	1.000000	225000.000000	9.000000	3.000000	0.000000	6000.000000
max	1.000000	450000.000000	27.000000	5.000000	5.000000	13500.000000

Fig.17 (Credit Limit)

Calculation of Profit per Customer:

The interest rate is set to 0.22. To calculate the PpC the credit limit is multiplied with interest rate and status.

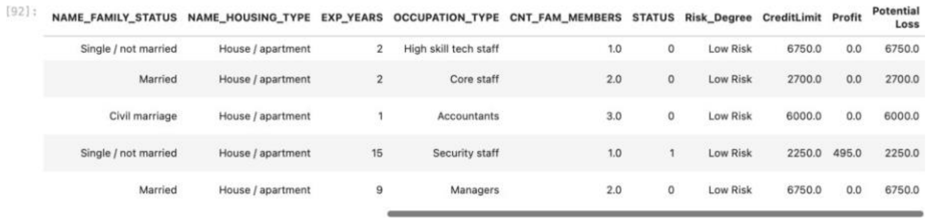


[90]:	FLAG_OWN_REALTY	AMT_INCOME_TOTAL	EXP_YEARS	CNT_FAM_MEMBERS	STATUS	CreditLimit	Profit
count	5355.000000	5355.000000	5355.000000	5355.000000	5355.000000	5355.000000	5355.000000
mean	0.348646	185222.368067	6.921569	2.297479	0.248179	4630.636359	203.810687
std	0.476586	81354.070365	5.557911	0.917509	0.634162	2734.840759	538.645394
min	0.000000	31500.000000	1.000000	1.000000	0.000000	-450.000000	-33.000000
25%	0.000000	135000.000000	3.000000	2.000000	0.000000	2850.000000	0.000000
50%	0.000000	166500.000000	5.000000	2.000000	0.000000	4050.000000	0.000000
75%	1.000000	225000.000000	9.000000	3.000000	0.000000	6000.000000	0.000000
max	1.000000	450000.000000	27.000000	5.000000	5.000000	13500.000000	7920.000000

Fig.18 (Profit Per Customer)

Calculation of Loss per Customer:

To calculate this it was assumed a proportion of customers will fail on their payments.



[92]:

NAME_FAMILY_STATUS	NAME_HOUSING_TYPE	EXP_YEARS	OCCUPATION_TYPE	CNT_FAM_MEMBERS	STATUS	Risk_Degree	CreditLimit	Profit	Potential Loss
Single / not married	House / apartment	2	High skill tech staff	1.0	0	Low Risk	6750.0	0.0	6750.0
Married	House / apartment	2	Core staff	2.0	0	Low Risk	2700.0	0.0	2700.0
Civil marriage	House / apartment	1	Accountants	3.0	0	Low Risk	6000.0	0.0	6000.0
Single / not married	House / apartment	15	Security staff	1.0	1	Low Risk	2250.0	495.0	2250.0
Married	House / apartment	9	Managers	2.0	0	Low Risk	6750.0	0.0	6750.0

Fig.19 (Loss per Customer)

Created a Bar plot between STATUS vs CREDITLIMIT

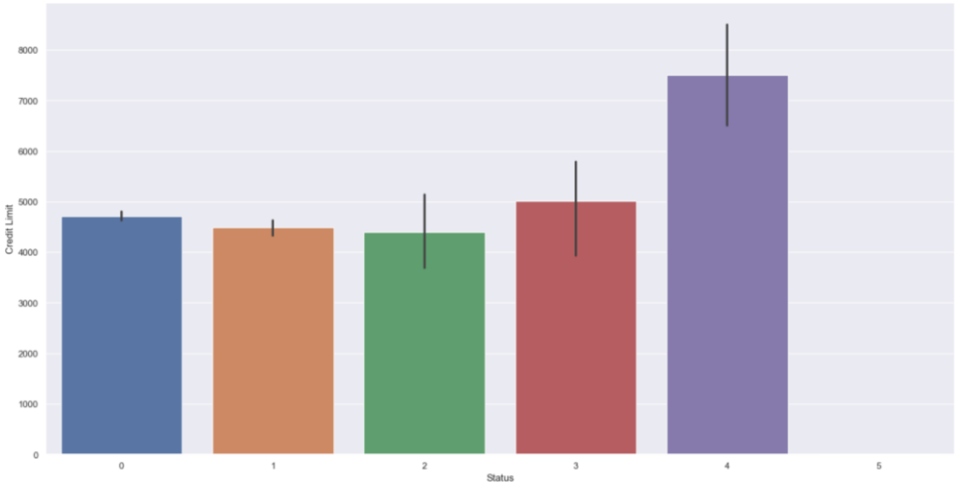


Fig. 20 (STATUS vs CREDIT LIMIT)

We have calculated the mean profit for each unique Risk_Degree.

Which clearly showed that the mean profit earned is very high and is around just over 5000 for the Moderate risk customers compared to the high risk customers which is ZERO.

The mean profit for the Low risk is also low and is just over ZERO.

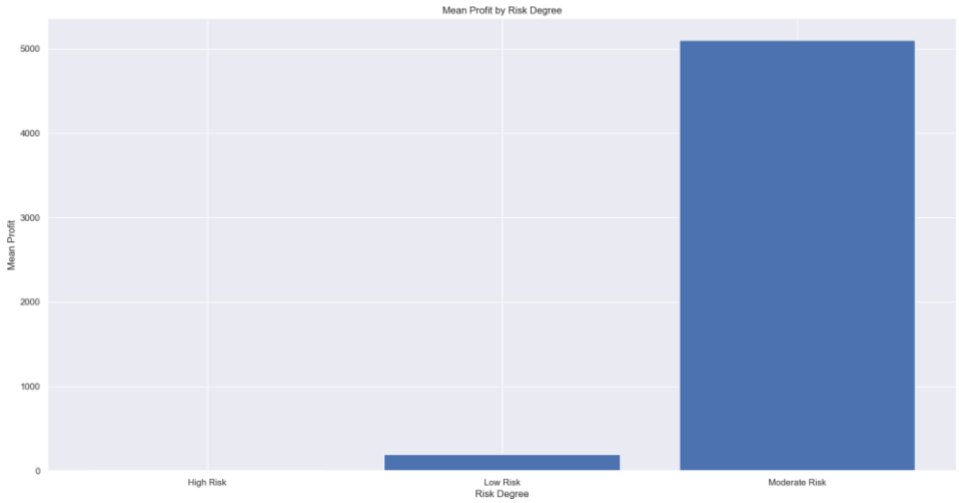


Fig.21 (Mean Profit by Risk Degree)

We have calculated the mean potential loss for each unique Risk_Degree.

Which clearly showed that the mean potential loss is very high and is around just over 6000 for the Moderate risk customers compared to the high risk customers which ZERO. The Potential loss of low-risk customerss is moderate and is just below 5000.

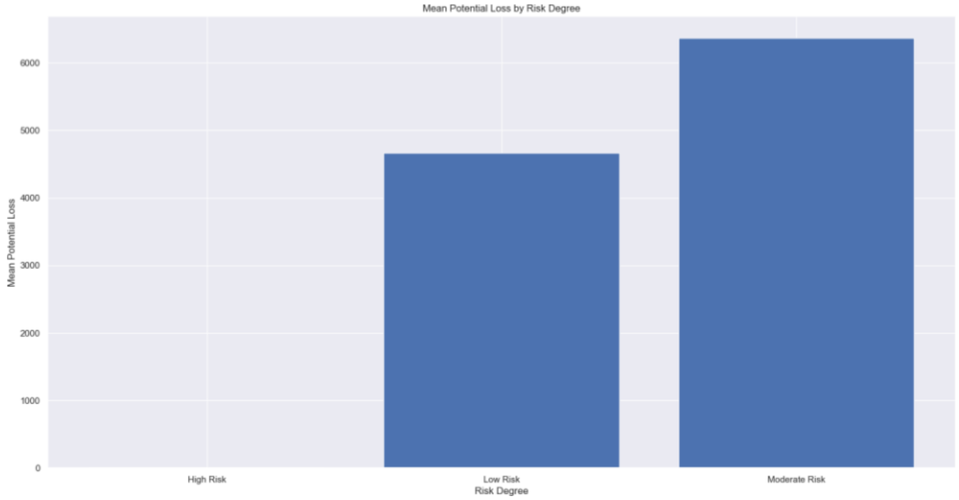


Fig.22 (Mean loss by Risk Degree)

To understand it better with our target variable STATUS, the mean potential profit and the mean potential loss graphs are below:

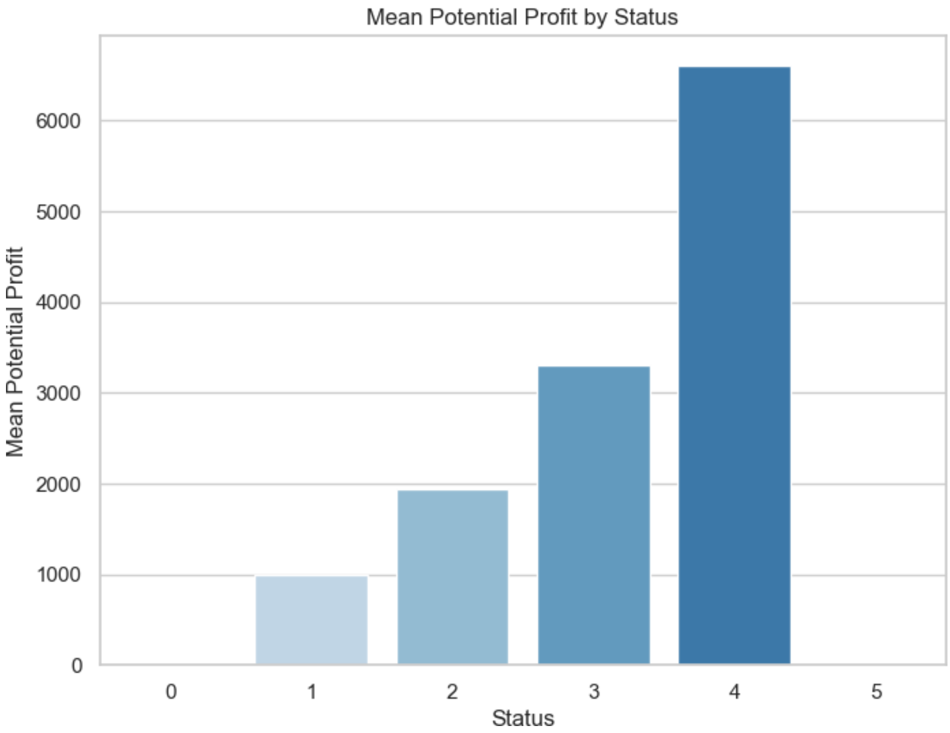


Fig.23 (Mean Profit by Status)

It is clear that the mean potential profit is the highest and is over 6000 for the STATUS 4.

Which is followed by the STATUS 3 with just over 3000, and STATUS 2 and STATUS 1 are below 2000.

The mean potential profit for the STATUS 5 ZERO.

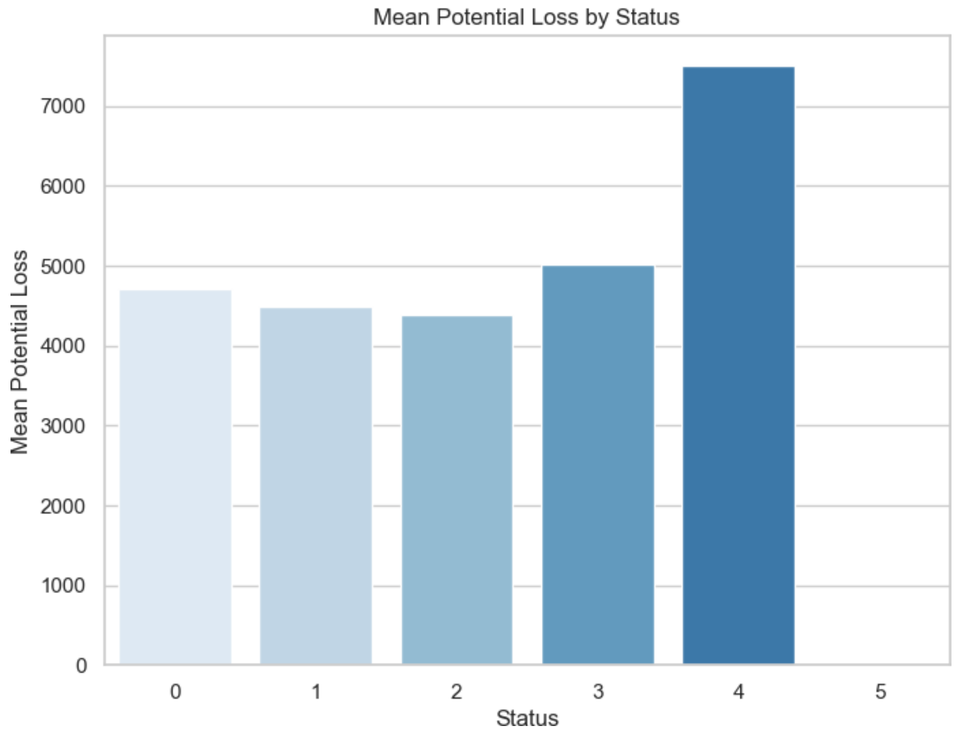


Fig.24 (Mean Loss by Status)

It is clear that the mean potential loss is again highest for the STATUS 4 and is over 7000.

Which is followed by the STATUS 0,1,2 and 3 which are quite simillar. The mean potential loss for the STATUS 3 is exactly 5000 and following the STATUS 0,1,2 is below 5000.

The mean potential loss for the STATUS 5 is again ZERO.

Discussion and Conclusions:

The Summary for the technical results:

Chi-Square Test: It was used to determine the relationship between different variables and the result. It was observed that the lower the p-value is the more significant is the relationship between the variable and the outcome.

Machine Learning Models: The 3 models were used for the prediction of the risk of credit card defaults: Random Forest, K-nearest neighbours (KNN), and Artificial Neural Network (ANN).

Random Forest: It showed an accuracy of 0.80, which is a good level of predictive performance. Overall it was robust to noise and overfitting and was much faster compared to KNN and ANN.

KNN: The accuracy performance was slightly better than Random Forest having 0.82. However, it was less robust to noise and overfitting compared to Random Forest.

ANN: It showed the least accuracy of 0.70, which showed it may not be the best model for this dataset.

Risk Assessment: The customers were categorized into low (0-2), moderate (3-4), and high risk (5) based on their statuses. This assigning the risk based on the status helps in better risk management.

Credit Limit Calculation: A credit limit was calculated for each customer based on their income and debt (debt-to-income ratio).

Profit and Loss Calculation: The potential profit and loss for each customer were calculated. From the results it was clear the potential profit is highest for moderate risk customers, while the potential loss was the highest for the same. From the analysis it as observed that a trade-off situation where moderate-risk customers could bring higher profits but also greater losses.

Correlation between Status and Credit Limit: A scatter plot was plotted to better understand the relationships between customers' status and their credit limit. This provided insights into how credit limit varies with the risk level.

Mean Potential Profit and Loss Analysis: After the analysis it was observed that the potential profit was highest for customers with STATUS 4 also the potential loss was highest for the same status. The STATUS 5 customers showed zero potential profit and loss, which showed that their credit activities are negligible or non-existent.

In conclusion, based on the accuracy, robustness, speed and interpretability, the Random Forest model was the most suitable for predicting credit card defaults in accordance to the business problem. The risk assessment method provided a strategy for credit limit assignment, and potential profit, and loss calculations.

To explain the results and solution to our manger we decided to dedicate a website for this process.

Where the potential customers can apply and provide the required details and our model will analyse the data they have provided and predict the customer falls under which risk categories and would it be beneficial in terms of earning profit for the bank or will it generate a loss for the bank when they approved the credit card.

After analyzing all the factors:

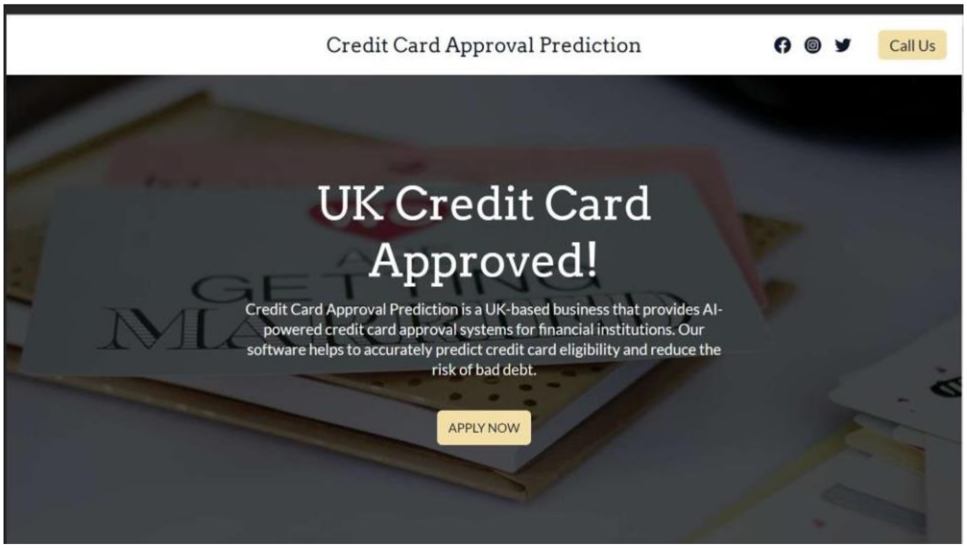
- Occupation Type
- FamilyStatus
- Own a Realty
- Work Experince In Years
- Income Type
- Income
- Education
- Own a Car
- Housing Type
- Family Members

Our model will give the:

STATUS – Approved or Not Approved

RISK DEGREE – High, Low or Moderate

CREDIT LIMIT – Which will be approved by taking in consideration all the above factors.

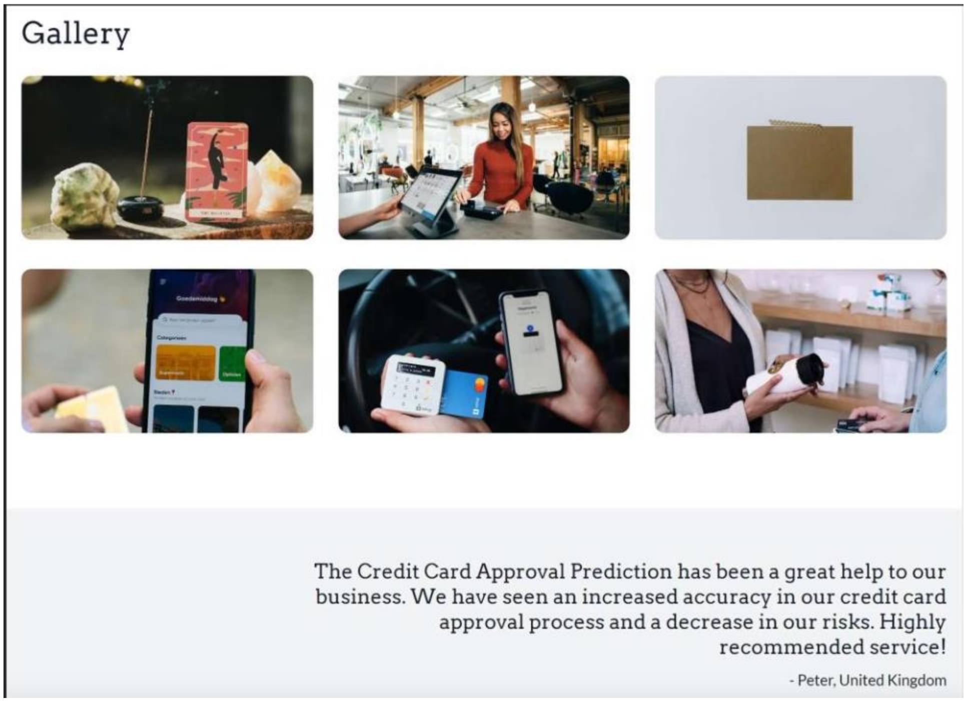


Credit Card Approval Prediction

UK Credit Card Approved!

Credit Card Approval Prediction is a UK-based business that provides AI-powered credit card approval systems for financial institutions. Our software helps to accurately predict credit card eligibility and reduce the risk of bad debt.

Fig. 25

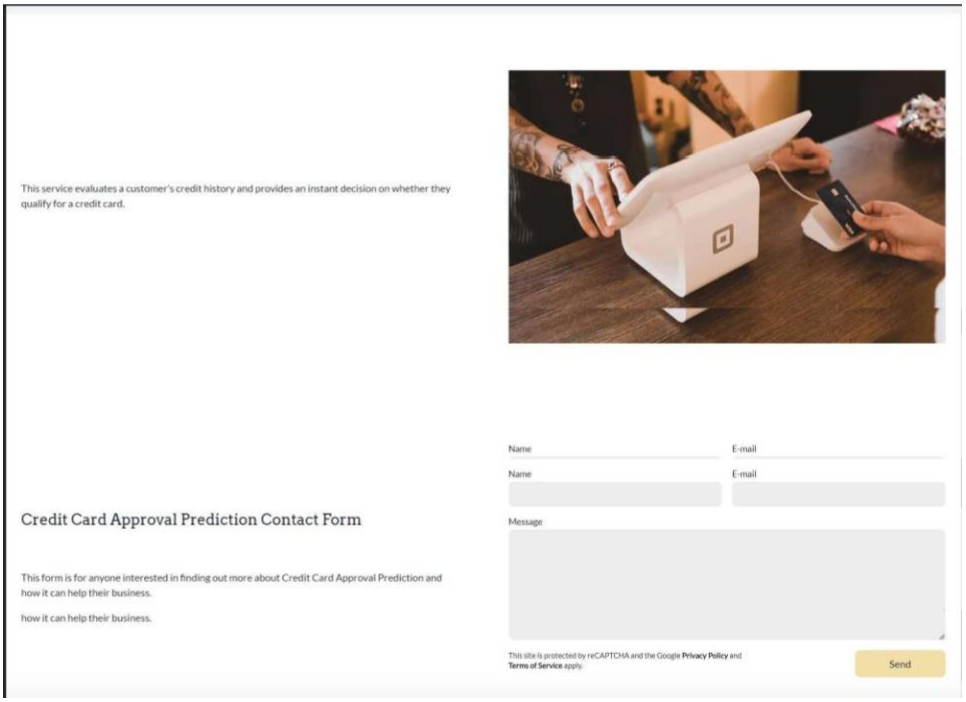


Gallery

The Credit Card Approval Prediction has been a great help to our business. We have seen an increased accuracy in our credit card approval process and a decrease in our risks. Highly recommended service!

- Peter, United Kingdom

Fig. 26



This service evaluates a customer's credit history and provides an instant decision on whether they qualify for a credit card.

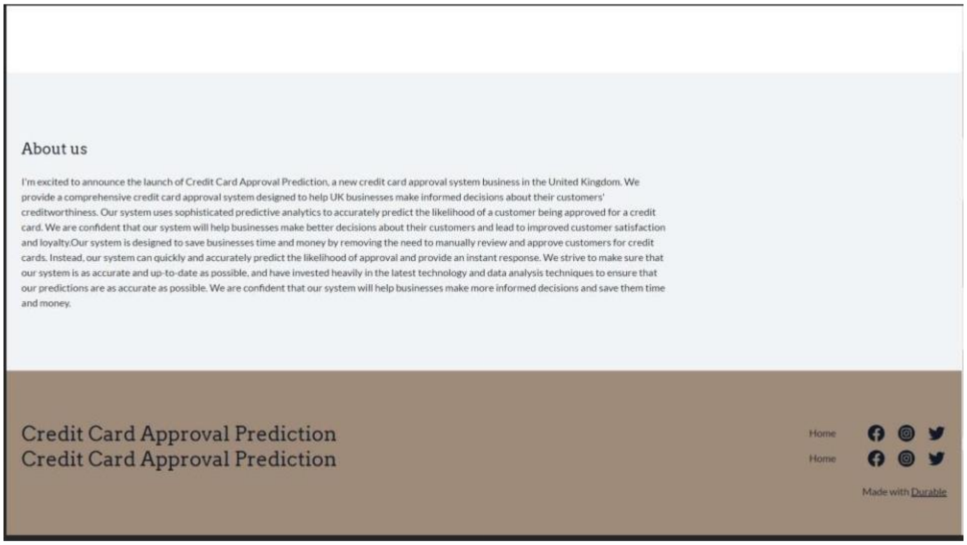
Credit Card Approval Prediction Contact Form

This form is for anyone interested in finding out more about Credit Card Approval Prediction and how it can help their business.

how it can help their business.

This site is protected by reCAPTCHA and the Google **Privacy Policy** and **Terms of Service** apply.

Fig. 27

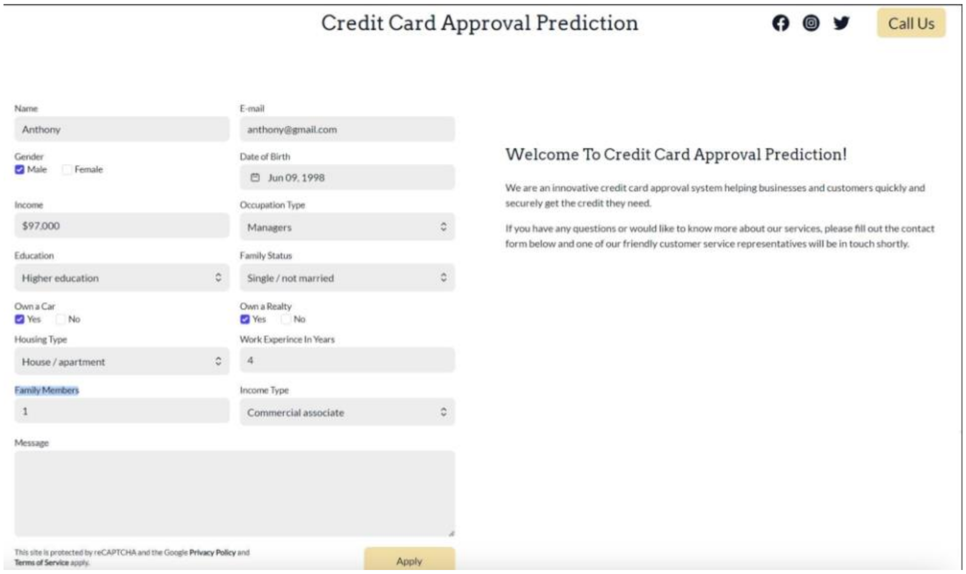


About us

I'm excited to announce the launch of Credit Card Approval Prediction, a new credit card approval system business in the United Kingdom. We provide a comprehensive credit card approval system designed to help UK businesses make informed decisions about their customers' creditworthiness. Our system uses sophisticated predictive analytics to accurately predict the likelihood of a customer being approved for a credit card. We are confident that our system will help businesses make better decisions about their customers and lead to improved customer satisfaction and loyalty.Our system is designed to save businesses time and money by removing the need to manually review and approve customers for credit cards. Instead, our system can quickly and accurately predict the likelihood of approval and provide an instant response. We strive to make sure that our system is as accurate and up-to-date as possible, and have invested heavily in the latest technology and data analysis techniques to ensure that our predictions are as accurate as possible. We are confident that our system will help businesses make more informed decisions and save them time and money.

Credit Card Approval Prediction
Credit Card Approval Prediction

Fig. 28



Credit Card Approval Prediction

Name: Anthony

E-mail: anthony@gmail.com

Gender: ☑ Male □ Female

Date of Birth: Jun 09, 1998

Income: $97,000

Occupation Type: Managers

Education: Higher education

Family Status: Single / not married

Own a Car: ☑ Yes □ No

Own a Realty: ☑ Yes □ No

Housing Type: House / apartment

Work Experince In Years: 4

Family Members: 1

Income Type: Commercial associate

Message

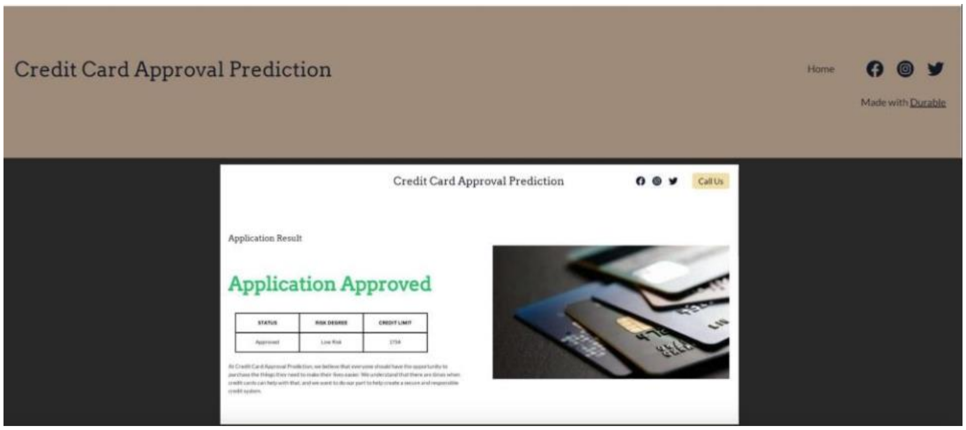
This site is protected by reCAPTCHA and the Google **Privacy Policy** and **Terms of Service** apply.

Welcome To Credit Card Approval Prediction!

We are an innovative credit card approval system helping businesses and customers quickly and securely get the credit they need.

If you have any questions or would like to know more about our services, please fill out the contact form below and one of our friendly customer service representatives will be in touch shortly.

Fig. 29



Credit Card Approval Prediction

Credit Card Approval Prediction

Application Result

Application Approved

STATUS	RISK DEGREE	CREDIT LIMIT
Approved	Low Risk	1758

At Credit Card Approval Prediction, we believe that everyone should have the opportunity to purchase the things they need to make their lives easier. We understand that there are times when credit cards can help with that, and we want to do our part to help create a secure and responsible credit system.

Fig. 25,26,27,28,29,30 (Credit Card Website)

For our non-technical audience:

The graphs STATUS vs CREDITLIMIT, Mean Profit vs Risk Degree, Mean Loss vs Risk Degree, Mean Loss vs STATUS and Mean Profit vs STATUS is mentioned for a better understanding of the risk factors.

The limitations we faced during handling the data:

Limited Variables: The analysis is based on a limited number of variables. It wasn't mentioned that the data provided was from which region. The data can be changed with accordance to the geographical regions. The provided data was Imbalanced Data. The more data can be collected which can differ by regions with more number of variables, which will help to analyse the data.

Organizational Changes Required:

Data Infrastructure: Collecting more data and managing it.

Technical Skillset: The organization needs to hire more employees to maintain and update their ML models.

Integration with Existing Systems: The current prediction model which is developed needs to be integrated with the current models which the organizations are using for better results.

Business Processes and Decisions Affected:

Risk Management: The prediction model could change how the organization manages risk where it can identify potentially risky customers easily and in advance to minimize the loss.

Customer Relationship Management: The organization may decide to engage with customers identified as high risk to prevent default to maintain a good relationship for the future.

Convincing Affected Parties:

Demonstrate Value: To demonstrate how the model can improve decision-making, reduce risk, and potentially increase profitability for the organizations.

Involve Stakeholders: The involvement of the stakeholders is very important, like credit officers, risk managers, in the development and implementation process to ensure buy-in.

REFRENCES

Awoyemi, J.O. (2017) *Credit card fraud detection using machine learning ... - IEEE xplore*. Available at: https://ieeexplore.ieee.org/abstract/document/8123782.

IJRASET (2022) *Implementation of credit card fraud detection using random forest algorithm*, *International Journal for Research in Applied Science and Engineering Technology*. Available at: https://www.academia.edu/73858709/Implementation_of_Credit_Card_Fraud_Detection_Using_Random_Forest_Algorithm.

Author links open overlay panelVincenzo Moscato *et al.* (2020) *A benchmark of machine learning approaches for Credit Score Prediction*, *Expert Systems with Applications*. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0957417420307636.

PYMNTS.com (2020) *How MasterCard uses AI to fight fraud and make better credit decisions*, *Pymnts.com*. Available at: https://www.pymnts.com/news/artificial-intelligence/2020/how-mastercard-uses-ai-to-fight-fraud-and-make-better-credit-decisions/.

Donnelly, C. (2017) *HSBC adopts cloud-first strategy to solving Big Data Business Problems: Computer Weekly*, *ComputerWeekly.com*. Available at: https://www.computerweekly.com/news/450418305/HSBC-adopts-cloud-first-strategy-to-solving-big-data-business-problems.

Machine learning helps payment services detect fraud (no date) *Business Class: Trends and Insights | American Express*. Available at: https://www.americanexpress.com/en-gb/business/trends-and-insights/articles/payment-services-fraud-detection-using-AI/ (Accessed: 10 May 2023).

Home (no date) *PayPal Newsroom*. Available at: https://www.paypal.com/stories/us/paypal-credit-risk-analysis-with-machine-learning. (Accessed: 10 May 2023).

Capital. Available at: https://www.ciodive.com/news/capital-one-puts-machine-learning-to-work-in-credit-decisions/539270/.

APPENDIX

```
In [1]: !pip install dataprep
        import csv
        import pandas as pd
        import numpy as np
        import seaborn as sns
        import matplotlib.pyplot as plt
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import accuracy_score
        from sklearn.preprocessing import OneHotEncoder, OrdinalEncoder
        from sklearn.base import BaseEstimator, TransformerMixin
        from sklearn.pipeline import Pipeline
        from sklearn.pipeline import FeatureUnion
        from sklearn.tree import DecisionTreeClassifier
        from sklearn.ensemble import GradientBoostingClassifier
        #from xgboost import XGBClassifier
        from sklearn.neighbors import KNeighborsClassifier
        from pandas import DataFrame
        import missingno as msno
        from dataprep.eda import create_report, plot, plot_correlation, plot_miss
```

```
Defaulting to user installation because normal site-packages is not writ
eable
Requirement already satisfied: dataprep in c:\users\myer\appdata\roaming
\python\python39\site-packages (0.4.5)
Requirement already satisfied: aiohttp<4.0,>=3.6 in c:\users\myer\appdat
a\roaming\python\python39\site-packages (from dataprep) (3.8.4)
Requirement already satisfied: pandas<2.0,>=1.1 in c:\programdata\anacon
da3\lib\site-packages (from dataprep) (1.4.4)
Requirement already satisfied: regex<2022.0.0,>=2021.8.3 in c:\users\mye
r\appdata\roaming\python\python39\site-packages (from dataprep) (2021.11
.10)
Requirement already satisfied: tqdm<5.0,>=4.48 in c:\programdata\anacond
a3\lib\site-packages (from dataprep) (4.64.1)
Requirement already satisfied: ipywidgets<8.0,>=7.5 in c:\programdata\an
aconda3\lib\site-packages (from dataprep) (7.6.5)
Requirement already satisfied: scipy<2.0,>=1.8 in c:\programdata\anacond
a3\lib\site-packages (from dataprep) (1.9.1)
Requirement already satisfied: dask[array,dataframe,delayed]>=2022.3.0 i
n c:\programdata\anaconda3\lib\site-packages (from dataprep) (2022.7.0)
Requirement already satisfied: jsonpath-ng<2.0,>=1.5 in c:\users\myer\ap
pdata\roaming\python\python39\site-packages (from dataprep) (1.5.3)
Requirement already satisfied: flask<3,>=2 in c:\users\myer\appdata\roam
ing\python\python39\site-packages (from dataprep) (2.2.3)
Requirement already satisfied: pydot<2.0.0,>=1.4.2 in c:\programdata\ana
conda3\lib\site-packages (from dataprep) (1.4.2)
Requirement already satisfied: wordcloud<2.0,>=1.8 in c:\users\myer\appd
ata\roaming\python\python39\site-packages (from dataprep) (1.8.2.2)
Requirement already satisfied: flask_cors<4.0.0,>=3.0.10 in c:\users\mye
r\appdata\roaming\python\python39\site-packages (from dataprep) (3.0.10)
Requirement already satisfied: metaphone<0.7,>=0.6 in c:\users\myer\appd
ata\roaming\python\python39\site-packages (from dataprep) (0.6)
Requirement already satisfied: jinja2<3.1,>=3.0 in c:\users\myer\appdata
\roaming\python\python39\site-packages (from dataprep) (3.0.3)
Requirement already satisfied: bokeh<3,>=2 in c:\programdata\anaconda3\l
ib\site-packages (from dataprep) (2.4.3)
```

```
Requirement already satisfied: pydantic<2.0,>=1.6 in c:\users\myer\appda
ta\roaming\python\python39\site-packages (from dataprep) (1.10.7)
Requirement already satisfied: rapidfuzz<3.0.0,>=2.1.2 in c:\users\myer\
appdata\roaming\python\python39\site-packages (from dataprep) (2.13.7)
Requirement already satisfied: nltk<4.0.0,>=3.6.7 in c:\programdata\anac
onda3\lib\site-packages (from dataprep) (3.7)
Requirement already satisfied: numpy<2.0,>=1.21 in c:\programdata\anacon
da3\lib\site-packages (from dataprep) (1.21.5)
Requirement already satisfied: python-stdnum<2.0,>=1.16 in c:\users\myer
\appdata\roaming\python\python39\site-packages (from dataprep) (1.18)
Requirement already satisfied: python-crfsuite==0.9.8 in c:\users\myer\a
ppdata\roaming\python\python39\site-packages (from dataprep) (0.9.8)
Requirement already satisfied: sqlalchemy==1.3.24 in c:\users\myer\appda
ta\roaming\python\python39\site-packages (from dataprep) (1.3.24)
Requirement already satisfied: varname<0.9.0,>=0.8.1 in c:\users\myer\ap
pdata\roaming\python\python39\site-packages (from dataprep) (0.8.3)
Requirement already satisfied: frozenlist>=1.1.1 in c:\users\myer\appdat
a\roaming\python\python39\site-packages (from aiohttp<4.0,>=3.6->datapre
p) (1.3.3)
Requirement already satisfied: charset-normalizer<4.0,>=2.0 in c:\progra
mdata\anaconda3\lib\site-packages (from aiohttp<4.0,>=3.6->dataprep) (2.
0.4)
Requirement already satisfied: yarl<2.0,>=1.0 in c:\users\myer\appdata\r
oaming\python\python39\site-packages (from aiohttp<4.0,>=3.6->dataprep)
(1.8.2)
Requirement already satisfied: attrs>=17.3.0 in c:\programdata\anaconda3
\lib\site-packages (from aiohttp<4.0,>=3.6->dataprep) (21.4.0)
Requirement already satisfied: multidict<7.0,>=4.5 in c:\users\myer\appd
ata\roaming\python\python39\site-packages (from aiohttp<4.0,>=3.6->datap
rep) (6.0.4)
Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in c:\users\m
yer\appdata\roaming\python\python39\site-packages (from aiohttp<4.0,>=3.
6->dataprep) (4.0.2)
Requirement already satisfied: aiosignal>=1.1.2 in c:\users\myer\appdata
\roaming\python\python39\site-packages (from aiohttp<4.0,>=3.6->dataprep
) (1.3.1)
Requirement already satisfied: pillow>=7.1.0 in c:\programdata\anaconda3
\lib\site-packages (from bokeh<3,>=2->dataprep) (9.2.0)
Requirement already satisfied: PyYAML>=3.10 in c:\programdata\anaconda3\
lib\site-packages (from bokeh<3,>=2->dataprep) (6.0)
Requirement already satisfied: packaging>=16.8 in c:\programdata\anacond
a3\lib\site-packages (from bokeh<3,>=2->dataprep) (21.3)
Requirement already satisfied: typing-extensions>=3.10.0 in c:\programda
ta\anaconda3\lib\site-packages (from bokeh<3,>=2->dataprep) (4.3.0)
Requirement already satisfied: tornado>=5.1 in c:\programdata\anaconda3\
lib\site-packages (from bokeh<3,>=2->dataprep) (6.1)
Requirement already satisfied: toolz>=0.8.2 in c:\programdata\anaconda3\
lib\site-packages (from dask[array,dataframe,delayed]>=2022.3.0->datapre
p) (0.11.2)
Requirement already satisfied: fsspec>=0.6.0 in c:\programdata\anaconda3
\lib\site-packages (from dask[array,dataframe,delayed]>=2022.3.0->datapr
ep) (2022.7.1)
Requirement already satisfied: cloudpickle>=1.1.1 in c:\programdata\anac
onda3\lib\site-packages (from dask[array,dataframe,delayed]>=2022.3.0->d
ataprep) (2.0.0)
Requirement already satisfied: partd>=0.3.10 in c:\programdata\anaconda3
\lib\site-packages (from dask[array,dataframe,delayed]>=2022.3.0->datapr
ep) (1.2.0)
Requirement already satisfied: Werkzeug>=2.2.2 in c:\users\myer\appdata\
roaming\python\python39\site-packages (from flask<3,>=2->dataprep) (2.2.
3)
Requirement already satisfied: importlib-metadata>=3.6.0 in c:\programda
ta\anaconda3\lib\site-packages (from flask<3,>=2->dataprep) (4.11.3)
Requirement already satisfied: itsdangerous>=2.0 in c:\programdata\anaco
nda3\lib\site-packages (from flask<3,>=2->dataprep) (2.0.1)
Requirement already satisfied: click>=8.0 in c:\programdata\anaconda3\li
b\site-packages (from flask<3,>=2->dataprep) (8.0.4)
Requirement already satisfied: Six in c:\programdata\anaconda3\lib\site-
packages (from flask_cors<4.0.0,>=3.0.10->dataprep) (1.16.0)
Requirement already satisfied: ipython>=4.0.0 in c:\programdata\anaconda
3\lib\site-packages (from ipywidgets<8.0,>=7.5->dataprep) (7.31.1)
Requirement already satisfied: nbformat>=4.2.0 in c:\programdata\anacond
a3\lib\site-packages (from ipywidgets<8.0,>=7.5->dataprep) (5.5.0)
Requirement already satisfied: traitlets>=4.3.1 in c:\programdata\anacon
da3\lib\site-packages (from ipywidgets<8.0,>=7.5->dataprep) (5.1.1)
Requirement already satisfied: ipykernel>=4.5.1 in c:\programdata\anacon
da3\lib\site-packages (from ipywidgets<8.0,>=7.5->dataprep) (6.15.2)
Requirement already satisfied: jupyterlab-widgets>=1.0.0 in c:\programda
ta\anaconda3\lib\site-packages (from ipywidgets<8.0,>=7.5->dataprep) (1.
0.0)
Requirement already satisfied: widgetsnbextension~=3.5.0 in c:\programda
ta\anaconda3\lib\site-packages (from ipywidgets<8.0,>=7.5->dataprep) (3.
5.2)
Requirement already satisfied: ipython-genutils~=0.2.0 in c:\programdata
\anaconda3\lib\site-packages (from ipywidgets<8.0,>=7.5->dataprep) (0.2.
0)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\myer\appdata\
roaming\python\python39\site-packages (from jinja2<3.1,>=3.0->dataprep)
(2.1.2)
Requirement already satisfied: ply in c:\users\myer\appdata\roaming\pyth
on\python39\site-packages (from jsonpath-ng<2.0,>=1.5->dataprep) (3.11)
Requirement already satisfied: decorator in c:\programdata\anaconda3\lib
\site-packages (from jsonpath-ng<2.0,>=1.5->dataprep) (5.1.1)
Requirement already satisfied: joblib in c:\programdata\anaconda3\lib\si
te-packages (from nltk<4.0.0,>=3.6.7->dataprep) (1.1.0)
Requirement already satisfied: pytz>=2020.1 in c:\programdata\anaconda3\
lib\site-packages (from pandas<2.0,>=1.1->dataprep) (2022.1)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\programdata\
anaconda3\lib\site-packages (from pandas<2.0,>=1.1->dataprep) (2.8.2)
Requirement already satisfied: pyparsing>=2.1.4 in c:\programdata\anacon
da3\lib\site-packages (from pydot<2.0.0,>=1.4.2->dataprep) (3.0.9)
Requirement already satisfied: colorama in c:\programdata\anaconda3\lib\
site-packages (from tqdm<5.0,>=4.48->dataprep) (0.4.5)
Requirement already satisfied: asttokens<3.0.0,>=2.0.0 in c:\users\myer\
appdata\roaming\python\python39\site-packages (from varname<0.9.0,>=0.8.
1->dataprep) (2.2.1)
Requirement already satisfied: pure_eval<1.0.0 in c:\users\myer\appdata\
roaming\python\python39\site-packages (from varname<0.9.0,>=0.8.1->datap
rep) (0.2.2)
Requirement already satisfied: executing<0.9.0,>=0.8.3 in c:\users\myer\
appdata\roaming\python\python39\site-packages (from varname<0.9.0,>=0.8.
1->dataprep) (0.8.3)
Requirement already satisfied: matplotlib in c:\programdata\anaconda3\li
b\site-packages (from wordcloud<2.0,>=1.8->dataprep) (3.5.2)
```

```
Requirement already satisfied: zipp>=0.5 in c:\programdata\anaconda3\lib
\site-packages (from importlib-metadata>=3.6.0->flask<3,>=2->dataprep) (
3.8.0)
Requirement already satisfied: debugpy>=1.0 in c:\programdata\anaconda3\
lib\site-packages (from ipykernel>=4.5.1->ipywidgets<8.0,>=7.5->dataprep
) (1.5.1)
Requirement already satisfied: matplotlib-inline>=0.1 in c:\programdata\
anaconda3\lib\site-packages (from ipykernel>=4.5.1->ipywidgets<8.0,>=7.5
->dataprep) (0.1.6)
Requirement already satisfied: psutil in c:\programdata\anaconda3\lib\si
te-packages (from ipykernel>=4.5.1->ipywidgets<8.0,>=7.5->dataprep) (5.9
.0)
Requirement already satisfied: jupyter-client>=6.1.12 in c:\programdata\
anaconda3\lib\site-packages (from ipykernel>=4.5.1->ipywidgets<8.0,>=7.5
->dataprep) (7.3.4)
Requirement already satisfied: pyzmq>=17 in c:\programdata\anaconda3\lib
\site-packages (from ipykernel>=4.5.1->ipywidgets<8.0,>=7.5->dataprep) (
23.2.0)
Requirement already satisfied: nest-asyncio in c:\programdata\anaconda3\
lib\site-packages (from ipykernel>=4.5.1->ipywidgets<8.0,>=7.5->dataprep
) (1.5.5)
Requirement already satisfied: jedi>=0.16 in c:\programdata\anaconda3\li
b\site-packages (from ipython>=4.0.0->ipywidgets<8.0,>=7.5->dataprep) (0
.18.1)
Requirement already satisfied: setuptools>=18.5 in c:\programdata\anacon
da3\lib\site-packages (from ipython>=4.0.0->ipywidgets<8.0,>=7.5->datapr
ep) (63.4.1)
Requirement already satisfied: pygments in c:\programdata\anaconda3\lib\
site-packages (from ipython>=4.0.0->ipywidgets<8.0,>=7.5->dataprep) (2.1
1.2)
Requirement already satisfied: backcall in c:\programdata\anaconda3\lib\
site-packages (from ipython>=4.0.0->ipywidgets<8.0,>=7.5->dataprep) (0.2
.0)
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.
0.0 in c:\programdata\anaconda3\lib\site-packages (from ipython>=4.0.0->
ipywidgets<8.0,>=7.5->dataprep) (3.0.20)
Requirement already satisfied: pickleshare in c:\programdata\anaconda3\l
ib\site-packages (from ipython>=4.0.0->ipywidgets<8.0,>=7.5->dataprep) (
0.7.5)
Requirement already satisfied: fastjsonschema in c:\programdata\anaconda
3\lib\site-packages (from nbformat>=4.2.0->ipywidgets<8.0,>=7.5->datapre
p) (2.16.2)
Requirement already satisfied: jsonschema>=2.6 in c:\programdata\anacond
a3\lib\site-packages (from nbformat>=4.2.0->ipywidgets<8.0,>=7.5->datapr
ep) (4.16.0)
Requirement already satisfied: jupyter_core in c:\programdata\anaconda3\
lib\site-packages (from nbformat>=4.2.0->ipywidgets<8.0,>=7.5->dataprep)
(4.11.1)
Requirement already satisfied: locket in c:\programdata\anaconda3\lib\si
te-packages (from partd>=0.3.10->dask[array,dataframe,delayed]>=2022.3.0
->dataprep) (1.0.0)
Requirement already satisfied: notebook>=4.4.1 in c:\programdata\anacond
a3\lib\site-packages (from widgetsnbextension~=3.5.0->ipywidgets<8.0,>=7
.5->dataprep) (6.4.12)
Requirement already satisfied: idna>=2.0 in c:\programdata\anaconda3\lib
\site-packages (from yarl<2.0,>=1.0->aiohttp<4.0,>=3.6->dataprep) (3.3)
Requirement already satisfied: fonttools>=4.22.0 in c:\programdata\anaco
```

```
nda3\lib\site-packages (from matplotlib->wordcloud<2.0,>=1.8->dataprep)
(4.25.0)
Requirement already satisfied: cycler>=0.10 in c:\programdata\anaconda3\
lib\site-packages (from matplotlib->wordcloud<2.0,>=1.8->dataprep) (0.11
.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\programdata\anaco
nda3\lib\site-packages (from matplotlib->wordcloud<2.0,>=1.8->dataprep)
(1.4.2)
Requirement already satisfied: parso<0.9.0,>=0.8.0 in c:\programdata\ana
conda3\lib\site-packages (from jedi>=0.16->ipython>=4.0.0->ipywidgets<8.
0,>=7.5->dataprep) (0.8.3)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.
14.0 in c:\programdata\anaconda3\lib\site-packages (from jsonschema>=2.6
->nbformat>=4.2.0->ipywidgets<8.0,>=7.5->dataprep) (0.18.0)
Requirement already satisfied: entrypoints in c:\programdata\anaconda3\l
ib\site-packages (from jupyter-client>=6.1.12->ipykernel>=4.5.1->ipywidg
ets<8.0,>=7.5->dataprep) (0.4)
Requirement already satisfied: pywin32>=1.0 in c:\programdata\anaconda3\
lib\site-packages (from jupyter_core->nbformat>=4.2.0->ipywidgets<8.0,>=
7.5->dataprep) (302)
Requirement already satisfied: prometheus-client in c:\programdata\anaco
nda3\lib\site-packages (from notebook>=4.4.1->widgetsnbextension~=3.5.0-
>ipywidgets<8.0,>=7.5->dataprep) (0.14.1)
Requirement already satisfied: argon2-cffi in c:\programdata\anaconda3\l
ib\site-packages (from notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywi
dgets<8.0,>=7.5->dataprep) (21.3.0)
Requirement already satisfied: Send2Trash>=1.8.0 in c:\programdata\anaco
nda3\lib\site-packages (from notebook>=4.4.1->widgetsnbextension~=3.5.0-
>ipywidgets<8.0,>=7.5->dataprep) (1.8.0)
Requirement already satisfied: terminado>=0.8.3 in c:\programdata\anacon
da3\lib\site-packages (from notebook>=4.4.1->widgetsnbextension~=3.5.0->
ipywidgets<8.0,>=7.5->dataprep) (0.13.1)
Requirement already satisfied: nbconvert>=5 in c:\programdata\anaconda3\
lib\site-packages (from notebook>=4.4.1->widgetsnbextension~=3.5.0->ipyw
idgets<8.0,>=7.5->dataprep) (6.4.4)
Requirement already satisfied: wcwidth in c:\programdata\anaconda3\lib\s
ite-packages (from prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0->ipython
>=4.0.0->ipywidgets<8.0,>=7.5->dataprep) (0.2.5)
Requirement already satisfied: beautifulsoup4 in c:\programdata\anaconda
3\lib\site-packages (from nbconvert>=5->notebook>=4.4.1->widgetsnbextens
ion~=3.5.0->ipywidgets<8.0,>=7.5->dataprep) (4.11.1)
Requirement already satisfied: defusedxml in c:\programdata\anaconda3\li
b\site-packages (from nbconvert>=5->notebook>=4.4.1->widgetsnbextension~
=3.5.0->ipywidgets<8.0,>=7.5->dataprep) (0.7.1)
Requirement already satisfied: mistune<2,>=0.8.1 in c:\programdata\anaco
nda3\lib\site-packages (from nbconvert>=5->notebook>=4.4.1->widgetsnbext
ension~=3.5.0->ipywidgets<8.0,>=7.5->dataprep) (0.8.4)
Requirement already satisfied: testpath in c:\programdata\anaconda3\lib\
site-packages (from nbconvert>=5->notebook>=4.4.1->widgetsnbextension~=3
.5.0->ipywidgets<8.0,>=7.5->dataprep) (0.6.0)
Requirement already satisfied: pandocfilters>=1.4.1 in c:\programdata\an
aconda3\lib\site-packages (from nbconvert>=5->notebook>=4.4.1->widgetsnb
extension~=3.5.0->ipywidgets<8.0,>=7.5->dataprep) (1.5.0)
Requirement already satisfied: nbclient<0.6.0,>=0.5.0 in c:\programdata\
anaconda3\lib\site-packages (from nbconvert>=5->notebook>=4.4.1->widgets
nbextension~=3.5.0->ipywidgets<8.0,>=7.5->dataprep) (0.5.13)
Requirement already satisfied: jupyterlab-pygments in c:\programdata\ana
```

```
conda3\lib\site-packages (from nbconvert>=5->notebook>=4.4.1->widgetsnbe
xtension~=3.5.0->ipywidgets<8.0,>=7.5->dataprep) (0.1.2)
Requirement already satisfied: bleach in c:\programdata\anaconda3\lib\si
te-packages (from nbconvert>=5->notebook>=4.4.1->widgetsnbextension~=3.5
.0->ipywidgets<8.0,>=7.5->dataprep) (4.1.0)
Requirement already satisfied: pywinpty>=1.1.0 in c:\programdata\anacond
a3\lib\site-packages (from terminado>=0.8.3->notebook>=4.4.1->widgetsnbe
xtension~=3.5.0->ipywidgets<8.0,>=7.5->dataprep) (2.0.2)
Requirement already satisfied: argon2-cffi-bindings in c:\programdata\an
aconda3\lib\site-packages (from argon2-cffi->notebook>=4.4.1->widgetsnbe
xtension~=3.5.0->ipywidgets<8.0,>=7.5->dataprep) (21.2.0)
Requirement already satisfied: cffi>=1.0.1 in c:\programdata\anaconda3\l
ib\site-packages (from argon2-cffi-bindings->argon2-cffi->notebook>=4.4.
1->widgetsnbextension~=3.5.0->ipywidgets<8.0,>=7.5->dataprep) (1.15.1)
Requirement already satisfied: soupsieve>1.2 in c:\programdata\anaconda3
\lib\site-packages (from beautifulsoup4->nbconvert>=5->notebook>=4.4.1->
widgetsnbextension~=3.5.0->ipywidgets<8.0,>=7.5->dataprep) (2.3.1)
Requirement already satisfied: webencodings in c:\programdata\anaconda3\
lib\site-packages (from bleach->nbconvert>=5->notebook>=4.4.1->widgetsnb
extension~=3.5.0->ipywidgets<8.0,>=7.5->dataprep) (0.5.1)
Requirement already satisfied: pycparser in c:\programdata\anaconda3\lib
\site-packages (from cffi>=1.0.1->argon2-cffi-bindings->argon2-cffi->not
ebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets<8.0,>=7.5->dataprep)
(2.21)
```

In [2]:

```
application_df = pd.read_csv('applications_with_status.csv')
application_df.head()
```

Out[2]:

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AM
0	5008804	M	Y	Y	0	
1	5008805	M	Y	Y	0	
2	500880G	M	Y	Y	0	
3	5008808	F	N	Y	0	
4	500880U	F	N	Y	0	

In [3]:

```
application_df.shape
```

Out[3]:

```
(36457, 20)
```

In [4]:

```
application_df.isnull()
```

Out[4]:

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AM
0	False	False	False	False	False	
1	False	False	False	False	False	
2	False	False	False	False	False	
3	False	False	False	False	False	
4	False	False	False	False	False	
...	...	...	...	...	...	
3C452	False	False	False	False	False	
3C453	False	False	False	False	False	
3C454	False	False	False	False	False	
3C455	False	False	False	False	False	
3C45C	False	False	False	False	False	

3G457 rows × 20 columns

In [5]:
```
application_df.isnull().sum()
```

Out[5]:
```
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        11323
CNT_FAM_MEMBERS            0
DEFAULT                    0
STATUS                     0
dtype: int64
```

In [6]:
```
percent_missing = application_df.isnull().sum() * 100 / len(application_d
missing_value_df = pd.DataFrame({'column_name': application_df.columns,
                                 'percent_missing': percent_missing})
```

In [7]:
```
missing_value_df
```

Out[7]:

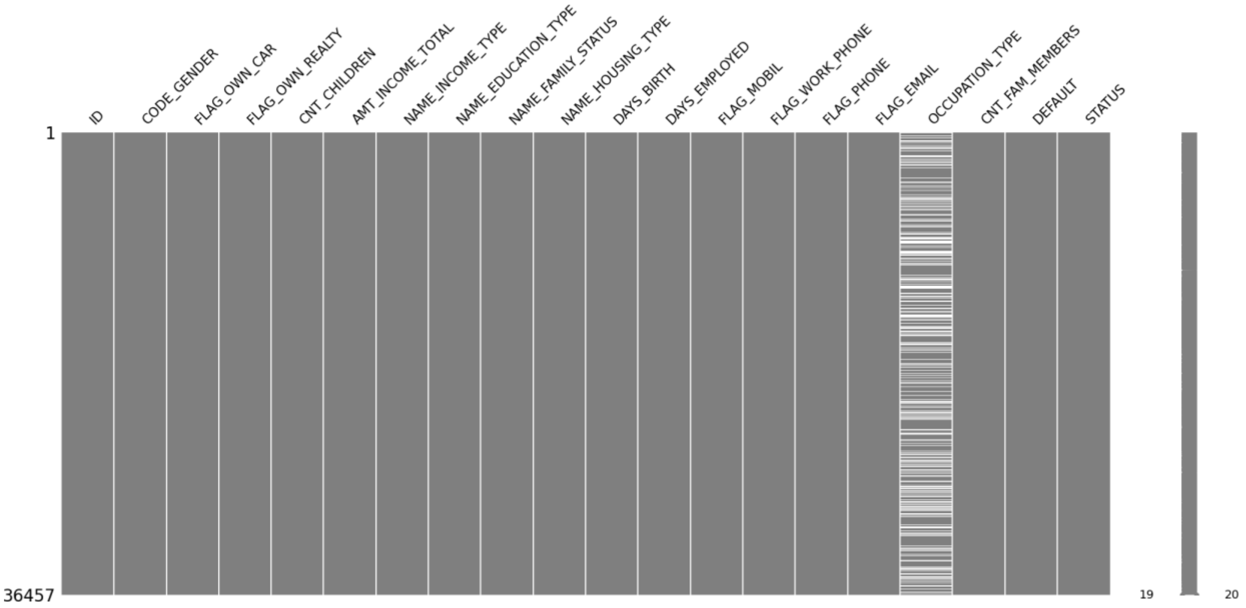
	column_name	percent_missing
ID	ID	0.000000
CODE_GENDER	CODE_GENDER	0.000000
FLAG_OWN_CAR	FLAG_OWN_CAR	0.000000
FLAG_OWN_REALTY	FLAG_OWN_REALTY	0.000000
CNT_CHILDREN	CNT_CHILDREN	0.000000
AMT_INCOME_TOTAL	AMT_INCOME_TOTAL	0.000000
NAME_INCOME_TYPE	NAME_INCOME_TYPE	0.000000
NAME_EDUCATION_TYPE	NAME_EDUCATION_TYPE	0.000000
NAME_FAMILY_STATUS	NAME_FAMILY_STATUS	0.000000
NAME_HOUSING_TYPE	NAME_HOUSING_TYPE	0.000000
DAYS_BIRTH	DAYS_BIRTH	0.000000
DAYS_EMPLOYED	DAYS_EMPLOYED	0.000000
FLAG_MOBIL	FLAG_MOBIL	0.000000
FLAG_WORK_PHONE	FLAG_WORK_PHONE	0.000000
FLAG_PHONE	FLAG_PHONE	0.000000
FLAG_EMAIL	FLAG_EMAIL	0.000000
OCCUPATION_TYPE	OCCUPATION_TYPE	31.058507
CNT_FAM_MEMBERS	CNT_FAM_MEMBERS	0.000000
DEFAULT	DEFAULT	0.000000
STATUS	STATUS	0.000000

In [8]:

```
msno.matrix(application_df, color = (0.5, 0.5, 0.5))
```

Out[8]:

```
<AxesSubplot:>
```



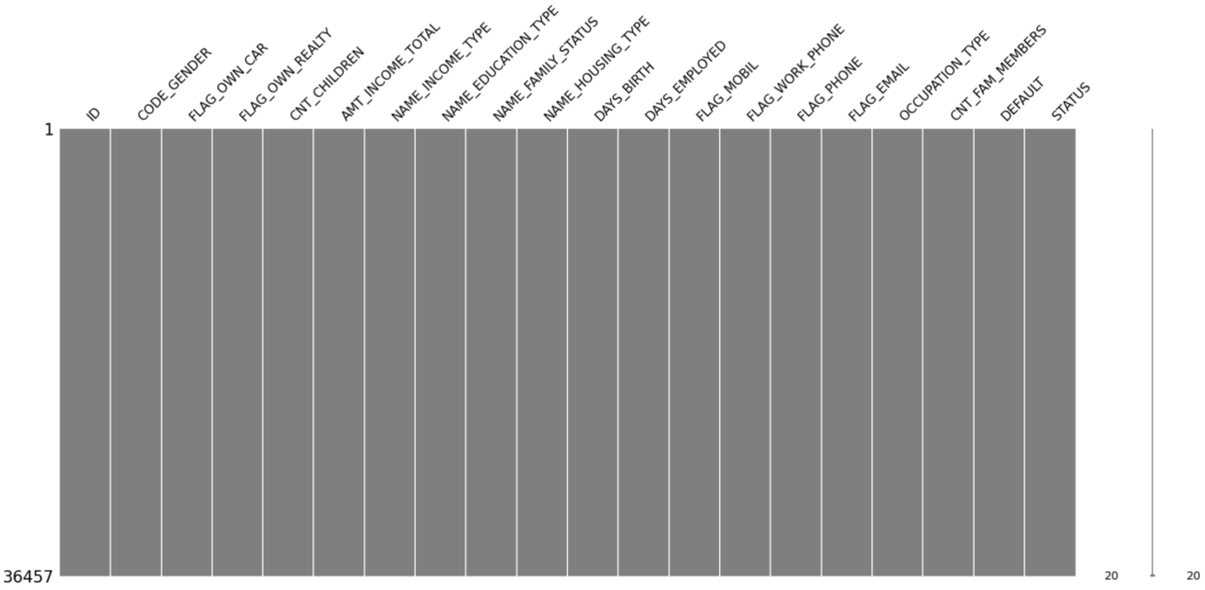
```
In [9]:  from pickle import TRUE
          application_df.fillna('Unspecified', inplace=True)
```

```
In [10]: # dropping null values
          application_df.dropna(inplace=True)
          application_df.shape
```

```
Out[10]: (36457, 20)
```

```
In [11]: msno.matrix(application_df, color = (0.5, 0.5, 0.5))
```

```
Out[11]: <AxesSubplot:>
```



In [12]:

```
# Create copy of df and convert columns into boolean
application_df['CODE_GENDER'].replace('M',0,inplace=True)
application_df['CODE_GENDER'].replace('F',1,inplace=True)
application_df['FLAG_OWN_CAR'].replace('Y',0,inplace=True)
application_df['FLAG_OWN_CAR'].replace('N',1,inplace=True)
application_df['FLAG_OWN_REALTY'].replace('Y',0,inplace=True)
application_df['FLAG_OWN_REALTY'].replace('N',1,inplace=True)
```

In [13]:

```
application_df['DAYS_BIRTH'] = application_df['DAYS_BIRTH'].abs()/365
application_df['DAYS_BIRTH'] = application_df['DAYS_BIRTH'].astype(int)
application_df['DAYS_EMPLOYED'] = application_df['DAYS_EMPLOYED'].abs()/3
application_df['DAYS_EMPLOYED'] = application_df['DAYS_EMPLOYED'].astype(
```

In [14]:

```
application_df.rename({'DAYS_BIRTH':'AGE', 'DAYS_EMPLOYED': 'EXP_YEARS'},
```

Exploratory Data Analysis

In [15]:

```
application_df.corr()
```

Out[15]:

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY
ID	1.000000	-0.012022	0.0111G3	0.0U8851
CODE_GENDER	-0.012022	1.000000	0.3G137U	-0.050758
FLAG_OWN_CAR	0.0111G3	0.3G137U	1.000000	-0.015185
FLAG_OWN_REALTY	0.0U8851	-0.050758	-0.015185	1.000000
CNT_CHILDREN	0.028878	-0.077GU0	-0.10583U	0.000575
AMT_INCOME_TOTAL	-0.017GG7	-0.1U7805	-0.21550G	-0.03271U
AGE	-0.05G03G	0.20224G	0.15G845	-0.12U344
EXP_YEARS	-0.038801	0.177071	0.158528	-0.0U3188
FLAG_MOBIL	NaN	NaN	NaN	NaN
FLAG_WORK_PHONE	0.07U215	-0.0G4UU4	-0.021G44	0.207732
FLAG_PHONE	0.00U87U	0.02G833	0.01401U	0.0GGG01
FLAG_EMAIL	-0.04GU7U	0.003284	-0.021750	-0.0521U4
CNT_FAM_MEMBERS	0.02GG24	-0.110782	-0.151814	0.005723
DEFAULT	0.010137	-0.00800G	0.00G731	0.005831

In [16]:

```
application_df.drop(columns=['ID','CNT_CHILDREN','FLAG_MOBIL','FLAG_WORK_
```

In [17]:

```
#application_df['STATUS'] = application_df["STATUS"].astype("string")
# application_df['FLAG_OWN_CAR'] = application_df['FLAG_OWN_CAR'].astype(
# application_df['FLAG_OWN_REALTY'] = application_df['FLAG_OWN_REALTY'].a
# application_df['FLAG_MOBIL'] = application_df['FLAG_MOBIL'].astype("boo
# application_df['FLAG_WORK_PHONE '] = application_df['FLAG_WORK_PHONE'].
# application_df['NAME_INCOME_TYPE'] = application_df['NAME_INCOME_TYPE']
# application_df['NAME_EDUCATION_TYPE'] = application_df['NAME_EDUCATION_
# application_df['NAME_FAMILY_STATUS'] = application_df['NAME_FAMILY_STAT
# application_df['NAME_HOUSING_TYPE'] = application_df['NAME_HOUSING_TYPE

application_df['STATUS'] = np.where((application_df['STATUS']=="X"), '0',
application_df['STATUS'] = application_df["STATUS"].astype("int64")
application_df['STATUS']
```

Out[17]:

```
0        1
1        1
2        0
3        0
4        0
        ..
36452    0
36453    0
36454    0
36455    0
36456    0
Name: STATUS, Length: 36457, dtype: int64
```

EXPLORATORY DATA ANALYSIS

In [18]:

```
application_df.corrwith(application_df['STATUS'])*100
```

Out[18]:

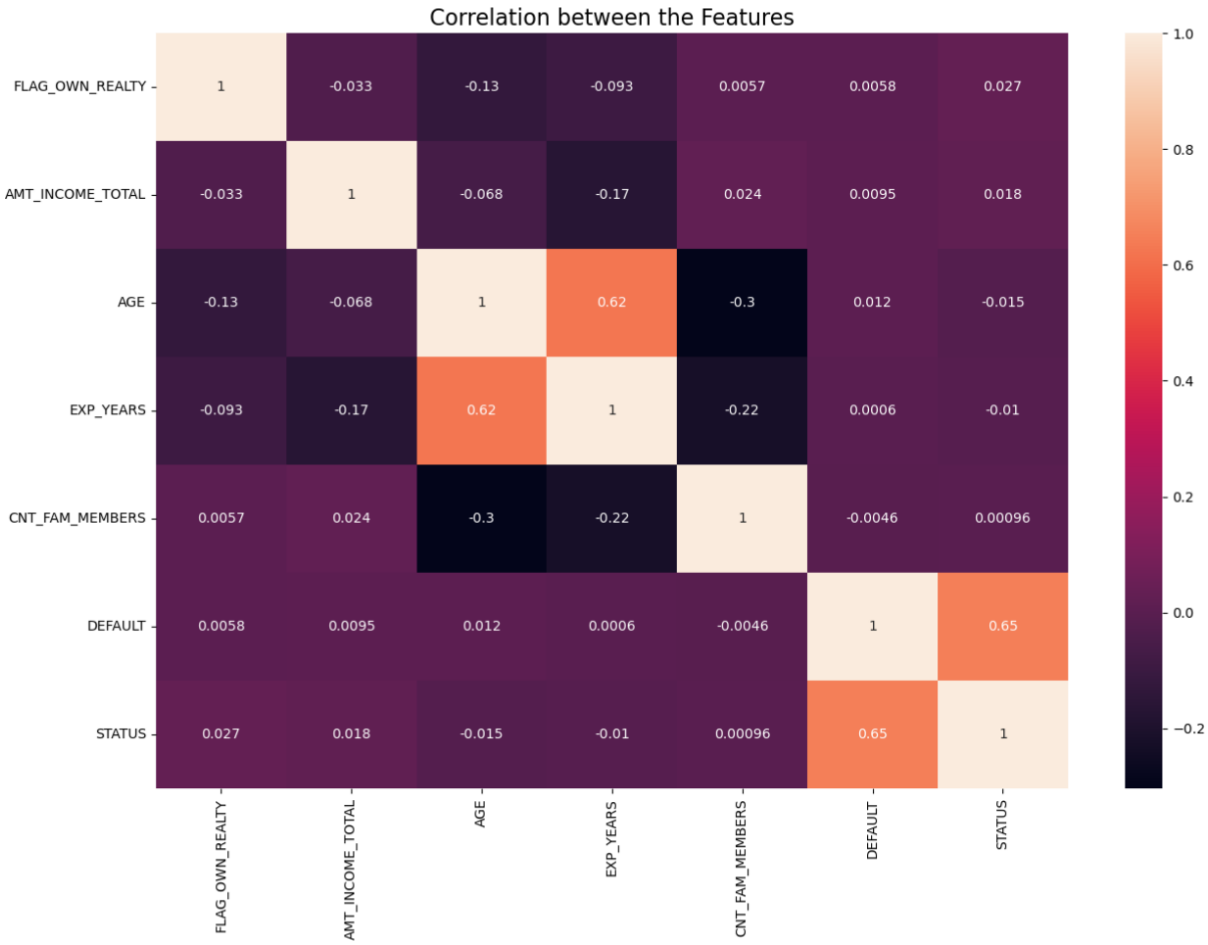
```
FLAG_OWN_REALTY       2.728038
AMT_INCOME_TOTAL      1.814709
AGE                  -1.474706
EXP_YEARS            -1.030182
CNT_FAM_MEMBERS       0.095926
DEFAULT              65.221820
STATUS              100.000000
dtype: float64
```

In [19]:

```
num_cols = [col for col in application_df.columns if application_df[col].
 str_cols = [col for col in application_df.columns if application_df[col].
```

In [20]:

```
plt.figure(figsize = (15,10))
sns.heatmap(application_df.corr(), annot = True )
plt.title('Correlation between the Features', size = 16)
plt.show()
```



In [21]:
```
application_df.columns
```

Out[21]:
```
Index(['FLAG_OWN_REALTY', 'AMT_INCOME_TOTAL', 'NAME_INCOME_TYPE',
        'NAME_EDUCATION_TYPE', 'NAME_FAMILY_STATUS', 'NAME_HOUSING_TYPE',
'AGE',
        'EXP_YEARS', 'OCCUPATION_TYPE', 'CNT_FAM_MEMBERS', 'DEFAULT', 'ST
ATUS'],
       dtype='object')
```

In [22]:

```
# chi-square Test

from scipy.stats import chi2_contingency

# Specify the variables to test
variables = [ 'FLAG_OWN_REALTY', 'AMT_INCOME_TOTAL',
       'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE', 'NAME_FAMILY_STATUS',
       'NAME_HOUSING_TYPE', 'EXP_YEARS',
       'OCCUPATION_TYPE', 'CNT_FAM_MEMBERS']

# Iterate through each variable and perform the chi-square test
for var in variables:
    # Create a contingency table of the variable and "STATUS"
    contingency_table = pd.crosstab(application_df[var], application_df["

    # Perform the chi-square test
    chi2, p, dof, expected = chi2_contingency(contingency_table)

    # Print the results
    print("Variable: ", var)
    print("Chi-square statistic: ", chi2)
    print("Degrees of freedom: ", dof)
    print("p-value: ", p)
    print("")
   ## print("Observed counts:")
   ## print(contingency_table)
   ## print("Expected counts:")
   ## print(pd.DataFrame(expected, index=contingency_table.index, columns
    ##print("")

    # # Create a stacked bar chart of the contingency table
   ## contingency_table.plot(kind="bar", stacked=True)
    ##plt.title("Relationship between " + var + " and STATUS")
    ##plt.xlabel(var)
   ## plt.ylabel("Count")
   ## plt.show()
```

```
Variable:  FLAG_OWN_REALTY
Chi-square statistic:  38.346836695541626
Degrees of freedom:  5
p-value:  3.213787377269442e-07

Variable:  AMT_INCOME_TOTAL
Chi-square statistic:  2562.898660575034
Degrees of freedom:  1320
p-value:  3.219044188068273e-82

Variable:  NAME_INCOME_TYPE
Chi-square statistic:  79.18948754900813
Degrees of freedom:  20
p-value:  5.386892848695465e-09

Variable:  NAME_EDUCATION_TYPE
Chi-square statistic:  62.024324230504064
Degrees of freedom:  20
p-value:  3.446100928540676e-06

Variable:  NAME_FAMILY_STATUS
Chi-square statistic:  107.6359528314522
Degrees of freedom:  20
p-value:  5.288861194414323e-14

Variable:  NAME_HOUSING_TYPE
Chi-square statistic:  49.168818105569216
Degrees of freedom:  25
p-value:  0.002694514926981239

Variable:  EXP_YEARS
Chi-square statistic:  297.6209401333129
Degrees of freedom:  220
p-value:  0.00037806153772468613

Variable:  OCCUPATION_TYPE
Chi-square statistic:  177.97235654787727
Degrees of freedom:  90
p-value:  9.704827413497206e-08

Variable:  CNT_FAM_MEMBERS
Chi-square statistic:  85.22137808677816
Degrees of freedom:  45
p-value:  0.0002750357678367596
```

In [23]:

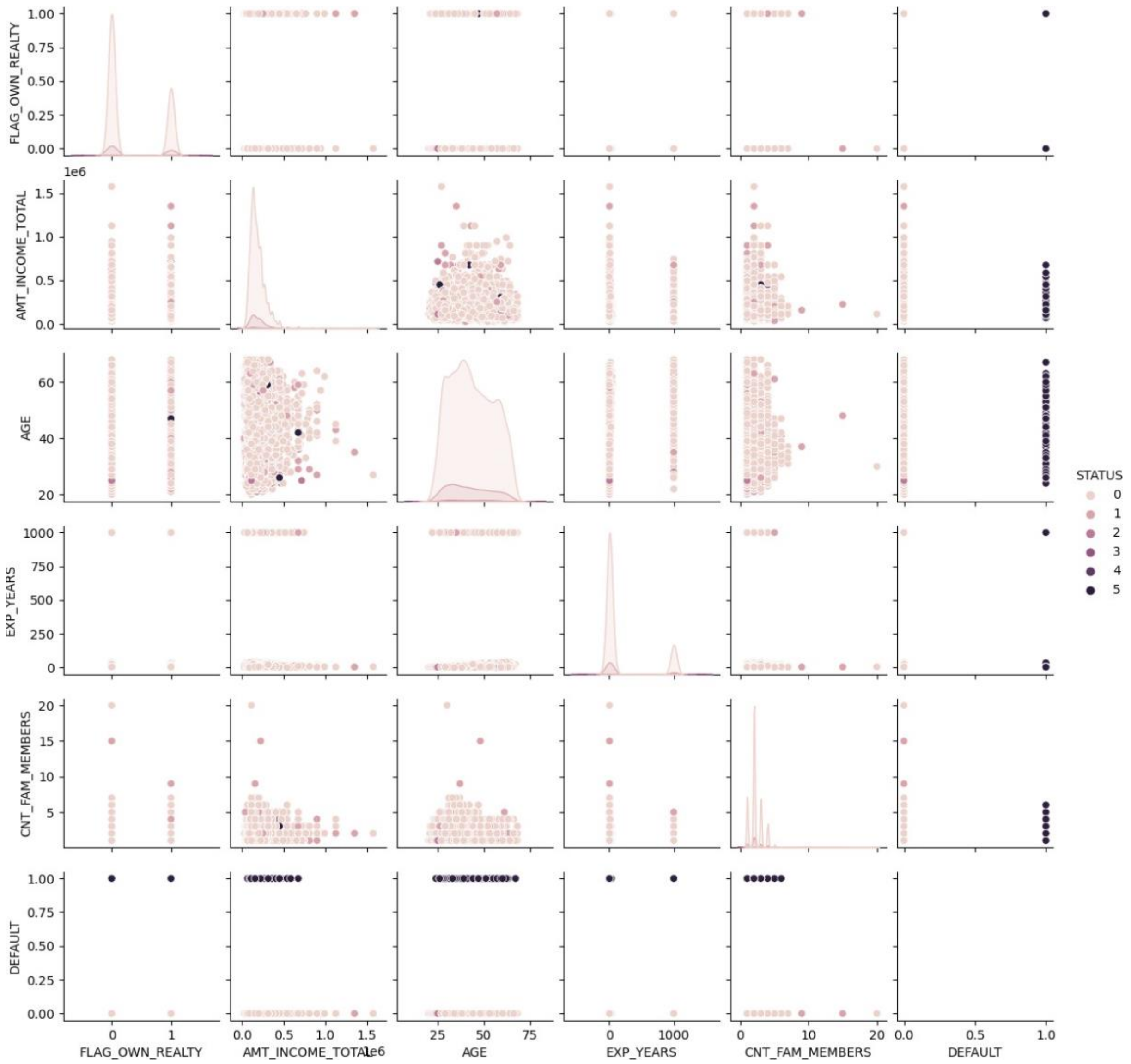
```
import plotly.graph_objs as go
import plotly.offline as py
%matplotlib inline
edu_type = application_df["NAME_EDUCATION_TYPE"].value_counts()
labels = (np.array(edu_type.index))
sizes = (np.array((edu_type / edu_type.sum())*100))

trace = go.Pie(labels=labels, values=sizes)
layout = go.Layout(title="Education Type")
dat = [trace]
fig = go.Figure(data=dat, layout=layout)
py.iplot(fig, filename="Education Type")
```

In [24]:

```
sns.pairplot(application_df, hue="STATUS", height=2)
```

Out[24]:

```
<seaborn.axisgrid.PairGrid at 0x2869cb94c70>
```



In [25]:
```
report = create_report(application_df)
```

```
  0%|
| 0/1521 [00:00<…
```

```
C:\Users\Myer\AppData\Roaming\Python\Python39\site-packages\dataprep\eda
\distribution\render.py:274: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in
a future version. Use pandas.concat instead.
```

In [26]:
```
report.show()
```

DataPrep Report Overview Variables ≡ Interactions Correla

Missing Values

Overview

Dataset Statistics

Number of Variables	12
Number of Rows	36457
Missing Cells	0
Missing Cells (%)	0.0%
Duplicate Rows	25353
Duplicate Rows (%)	6U.5%
Total Size in Memory	16.0 MB
Average Row Size in Memory	45U.5 B
Variable Types	Categorical: 8 Numerical: 4

Dataset Insights

AMT_INCOME_TOTAL is skewed

EXP_YEARS is skewed

CNT_FAM_MEMBERS is skewed

Dataset has 25353 (C9.54%) duplicate rows

FLAG_OWN_REALTY has constant length 1

STATUS has constant length 1

EXP_YEARS has 2540 (C.97%) zeros

Variables

Sort by Feature order □ Reverse order

FLAG_OWN_REALTY

categorical

Show Details

Approximate Distinct Count	2
Approximate Unique (%)	0.0%
Missing	0
Missing (%)	0.0%
Memory Size	2.3 MB

AMT_INCOME_TOTAL
numerical

Show Details

Approximate Distinct Count	265
Approximate Unique (%)	0.7%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Memory Size	56U.6 KB
Mean	186685.7367
Minimum	27000
Maximum	1.575e+06
Zeros	0
Zeros (%)	0.0%
Negatives	0
Negatives (%)	0.0%

NAME_INCOME_TYPE
categorical

Show Details

Approximate Distinct Count	5
Approximate Unique (%)	0.0%
Missing	0
Missing (%)	0.0%
Memory Size	2.6 MB

NAME_EDUCATION_...
categorical

Show Details

Approximate Distinct Count	5
Approximate Unique (%)	0.0%
Missing	0
Missing (%)	0.0%
Memory Size	3.1 MB

NAME_FAMILY_STA...
categorical

Show Details

Approximate Distinct Count	5
Approximate Unique (%)	0.0%
Missing	0
Missing (%)	0.0%
Memory Size	2.6 MB

NAME_HOUSING_T...
categorical

Show Details

Approximate Distinct Count	6
Approximate Unique (%)	0.0%
Missing	0
Missing (%)	0.0%
Memory Size	2.8 MB

AGE
numerical

Show Details

Approximate Distinct Count	4U
Approximate Unique (%)	0.1%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Memory Size	427.2 KB
Mean	43.2603
Minimum	20
Maximum	68
Zeros	0
Zeros (%)	0.0%
Negatives	0
Negatives (%)	0.0%

EXP_YEARS
numerical

Show Details

Approximate Distinct Count	45
Approximate Unique (%)	0.1%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Memory Size	427.2 KB
Mean	173.8U5
Minimum	0
Maximum	1000
Zeros	2540
Zeros (%)	7.0%
Negatives	0
Negatives (%)	0.0%

OCCUPATION_TYPE
categorical

Show Details

Approximate Distinct Count	1U
Approximate Unique (%)	0.1%
Missing	0
Missing (%)	0.0%
Memory Size	2.6 MB

CNT_FAM_MEMBERS
numerical

Show Details

Approximate Distinct Count	10
Approximate Unique (%)	0.0%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Memory Size	56U.6 KB
Mean	2.1U85
Minimum	1
Maximum	20
Zeros	0
Zeros (%)	0.0%
Negatives	0
Negatives (%)	0.0%

DEFAULT
categorical

Show Details

Approximate Distinct Count	2
Approximate Unique (%)	0.0%
Missing	0
Missing (%)	0.0%
Memory Size	2.4 MB

STATUS
categorical

Show Details

Approximate Distinct Count	6
Approximate Unique (%)	0.0%
Missing	0
Missing (%)	0.0%
Memory Size	2.3 MB

Interactions

Correlations

Pearson | Spearman | KendallTau

Missing Values

Bar Chart | Spectrum | Heat Map | Dendrogra

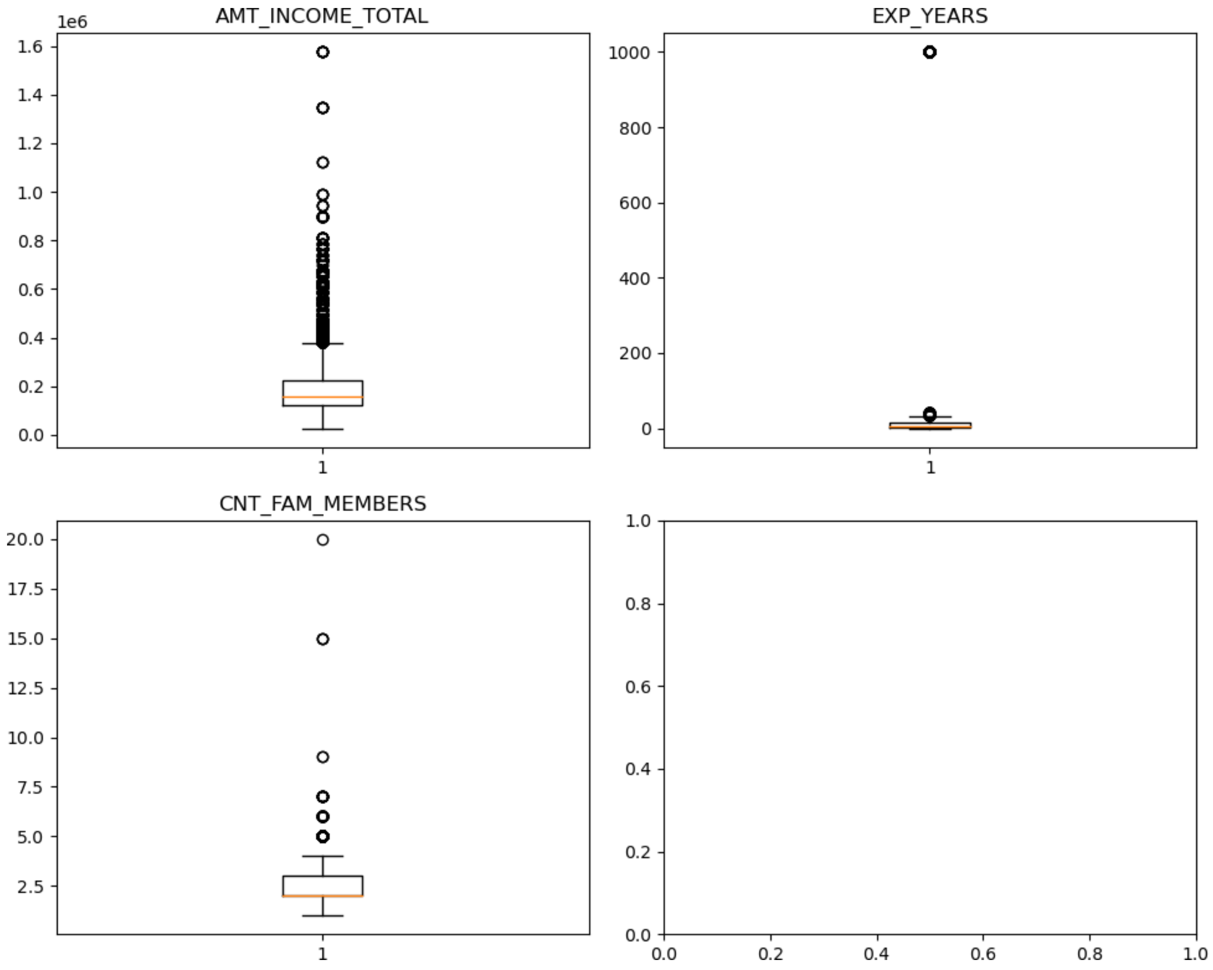
Report generated with DataPrep

In [27]:
```
application_df.describe()
```

Out[27]:

	FLAG_OWN_REALTY	AMT_INCOME_TOTAL	AGE	EXP_YEARS	CNT_FA
count	3G457.000000	3.G45700e+04	3G457.000000	3G457.000000	3
mean	0.327811	1.8GG857e+05	43.2G0334	173.8U5000	
std	0.4GU422	1.0178U2e+05	11.510414	371.G41572	
min	0.000000	2.700000e+04	20.000000	0.000000	
25%	0.000000	1.215000e+05	34.000000	3.000000	
50%	0.000000	1.575000e+05	42.000000	G.000000	
75%	1.000000	2.250000e+05	53.000000	15.000000	
max	1.000000	1.575000e+0G	G8.000000	1000.000000	

In [28]:
```
import matplotlib.pyplot as plt

# define the numerical columns
numerical_columns = ['AMT_INCOME_TOTAL', 'EXP_YEARS', 'CNT_FAM_MEMBERS']

# create a 2x2 grid of subplots
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(10, 8))

# create a box plot for each column in the DataFrame
for i, ax in enumerate(axes.flatten()):
    if i < len(numerical_columns):
        col = numerical_columns[i]
        ax.boxplot(application_df[col])
        ax.set_title(col)

# adjust the spacing between subplots
plt.tight_layout()

# display the plot
plt.show()
```



In [29]:

```
import numpy as np

# define the numerical columns
numerical_columns = ['AMT_INCOME_TOTAL', 'AGE', 'EXP_YEARS', 'CNT_FAM_MEM

# iterate over numerical columns and find outliers
for col in numerical_columns:
    Q1 = np.log(application_df[col]).quantile(0.25)
    Q3 = np.log(application_df[col]).quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 2.5 * IQR
    upper_bound = Q3 + 2.5 * IQR

    # identify outliers, take the log of them, and print the count of out
    outliers = application_df[(np.log(application_df[col]) < lower_bound)
    outliers_count_before = len(outliers)
    print(f"{col} outliers before removal: {outliers_count_before}")

    # remove outliers and print the count of outliers after removal
    application_df = application_df[(np.log(application_df[col]) >= lower
    outliers = application_df[(np.log(application_df[col]) < lower_bound)
    outliers_count_after = len(outliers)
    print(f"{col} outliers after removal: {outliers_count_after}")
```

```
AMT_INCOME_TOTAL outliers before removal: 17
AMT_INCOME_TOTAL outliers after removal: 0
AGE outliers before removal: 0
AGE outliers after removal: 0
EXP_YEARS outliers before removal: 8675
EXP_YEARS outliers after removal: 0
CNT_FAM_MEMBERS outliers before removal: 6
CNT_FAM_MEMBERS outliers after removal: 0
```

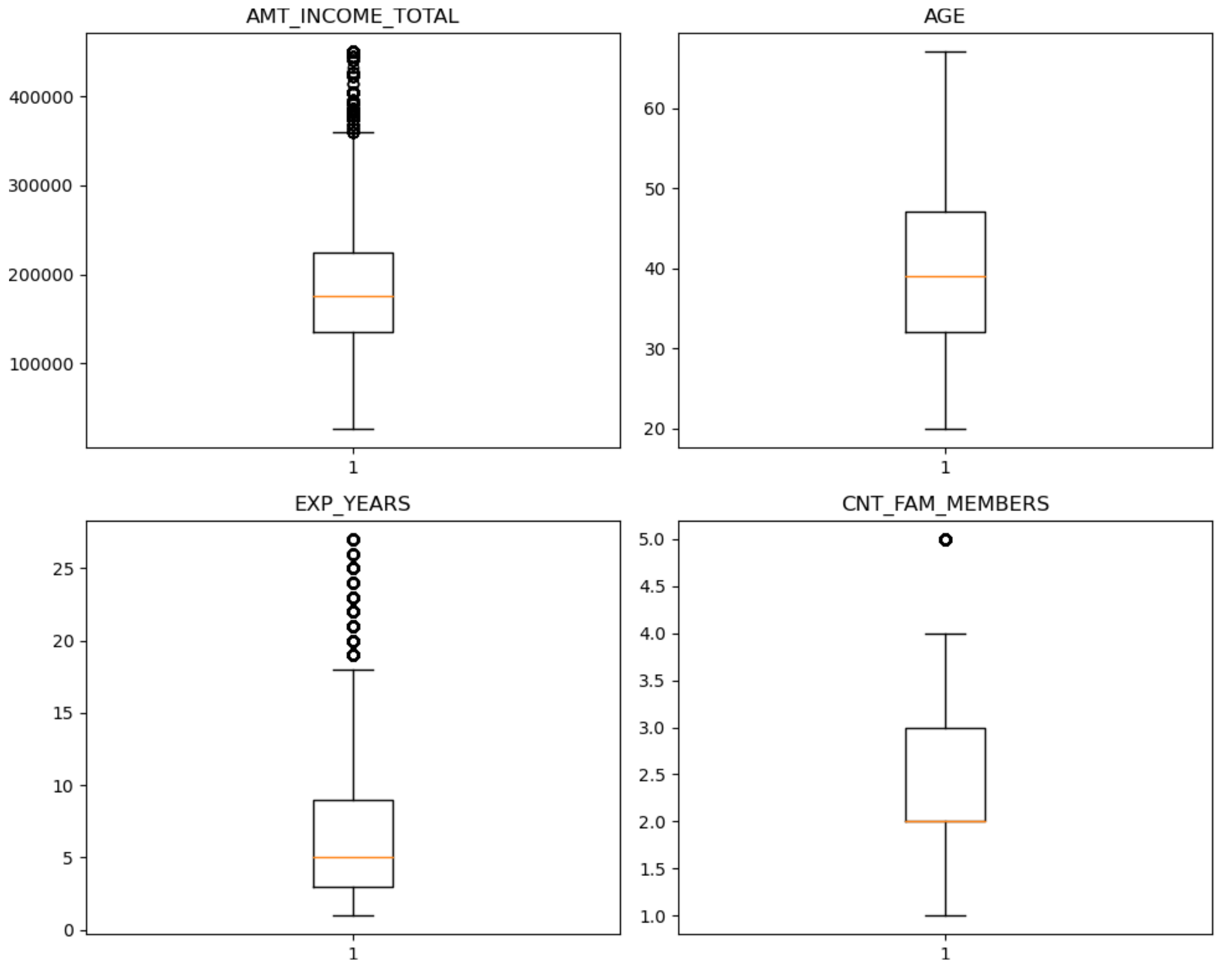
```
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\arraylike.py:397:
RuntimeWarning:

divide by zero encountered in log
```

In [30]:

```
import numpy as np

# define the numerical columns
numerical_columns = ['AMT_INCOME_TOTAL', 'AGE', 'EXP_YEARS', 'CNT_FAM_MEM

# create a new DataFrame for outliers
outliers_df = pd.DataFrame()

# iterate over numerical columns and find outliers
for col in numerical_columns:
    Q1 = application_df[col].quantile(0.25)
    Q3 = application_df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 2.5 * IQR
    upper_bound = Q3 + 2.5 * IQR

    # identify outliers and take the logarithm of their values
    outliers_mask = (application_df[col] < lower_bound) | (application_df
    outliers_df[col] = np.log(application_df[outliers_mask][col])

    # remove outliers from the original DataFrame
    application_df = application_df[~outliers_mask]
```

In [31]:

```
import matplotlib.pyplot as plt

# define the numerical columns
numerical_columns = ['AMT_INCOME_TOTAL', 'AGE', 'EXP_YEARS', 'CNT_FAM_MEM

# create a 2x2 grid of subplots
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(10, 8))

# create a box plot for each column in the DataFrame
for i, ax in enumerate(axes.flatten()):
    if i < len(numerical_columns):
        col = numerical_columns[i]
        ax.boxplot(application_df[col])
        ax.set_title(col)

# adjust the spacing between subplots
plt.tight_layout()

# display the plot
plt.show()
```



```
In [32]: application_df.describe()
```

Out[32]:

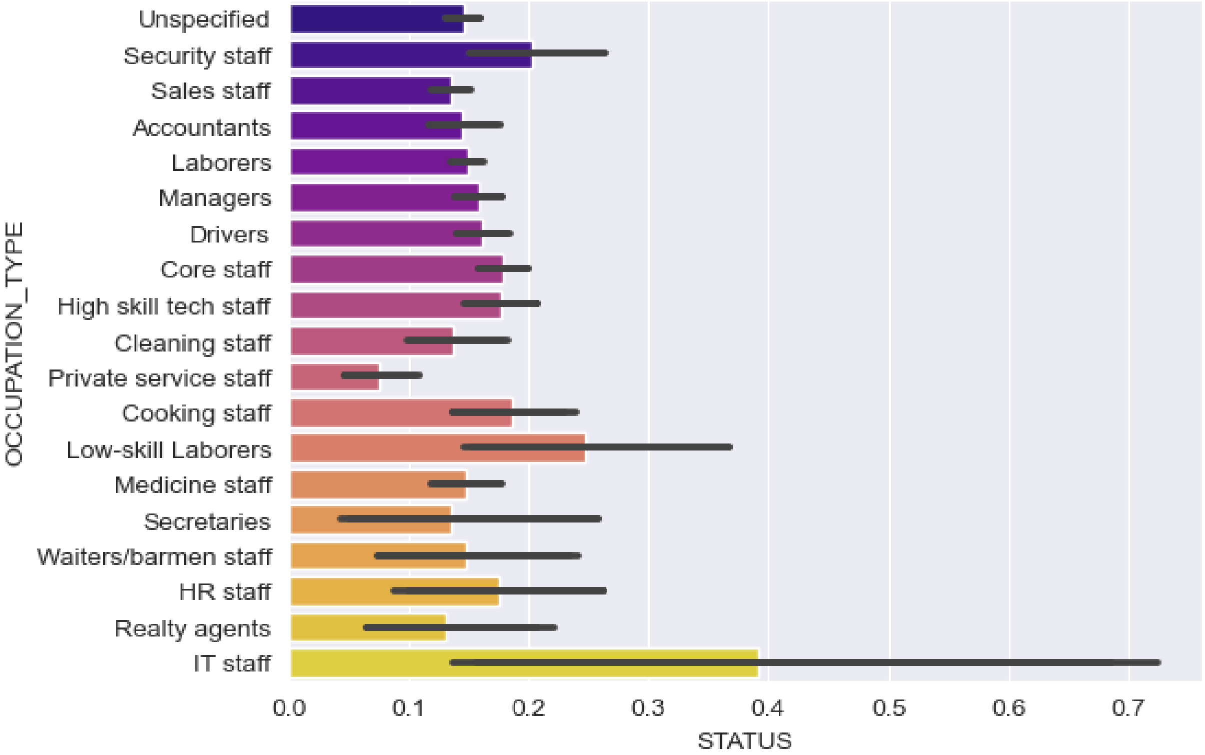
	FLAG_OWN_REALTY	AMT_INCOME_TOTAL	AGE	EXP_YEARS	CNT_FA
count	2G775.000000	2G775.000000	2G775.000000	2G775.000000	2
mean	0.348833	1872G2.157G47	3U.U85GUG	G.U42185	
std	0.47GG10	82250.548701	U.4GG401	5.582070	
min	0.000000	27000.000000	20.000000	1.000000	
25%	0.000000	135000.000000	32.000000	3.000000	
50%	0.000000	175500.000000	3U.000000	5.000000	
75%	1.000000	225000.000000	47.000000	U.000000	
max	1.000000	450000.000000	G7.000000	27.000000	

In [33]:

```
import seaborn as sns
# set the background style of the plot
sns.set_style('darkgrid')

# plot the graph using the default estimator mean
sns.barplot(x ='STATUS', y ='OCCUPATION_TYPE', data = application_df, pal

# or
import numpy as np

# change the estimator from mean to standard deviation
sns.barplot(x ='STATUS', y ='OCCUPATION_TYPE', data = application_df,
            palette ='plasma')
```

Out[33]:

```
<AxesSubplot:xlabel='STATUS', ylabel='OCCUPATION_TYPE'>
```

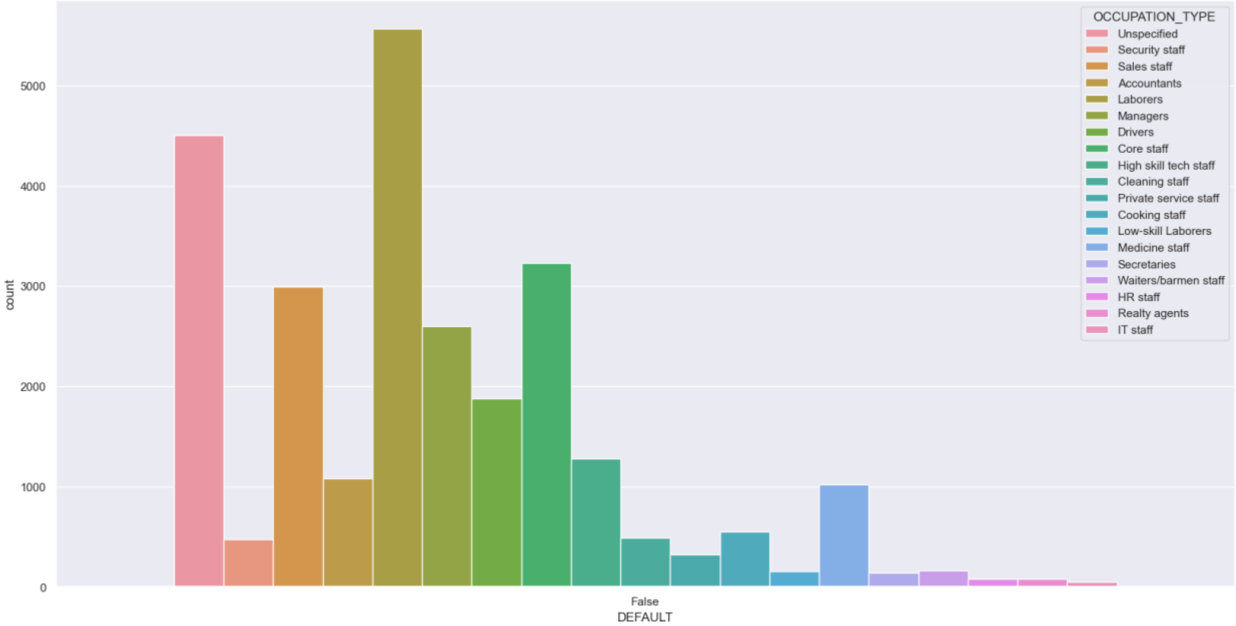


In [34]:

```
Target1 = application_df.loc[application_df["DEFAULT"]==True]
Target0 = application_df.loc[application_df["DEFAULT"]==False]
sns.set(rc={'figure.figsize':(20,10)})
#sns.countplot(data=Target1, x="DEFAULT", hue=application_df['OCCUPATION_
sns.countplot(data=Target0, x="DEFAULT", hue=application_df['OCCUPATION_T
```

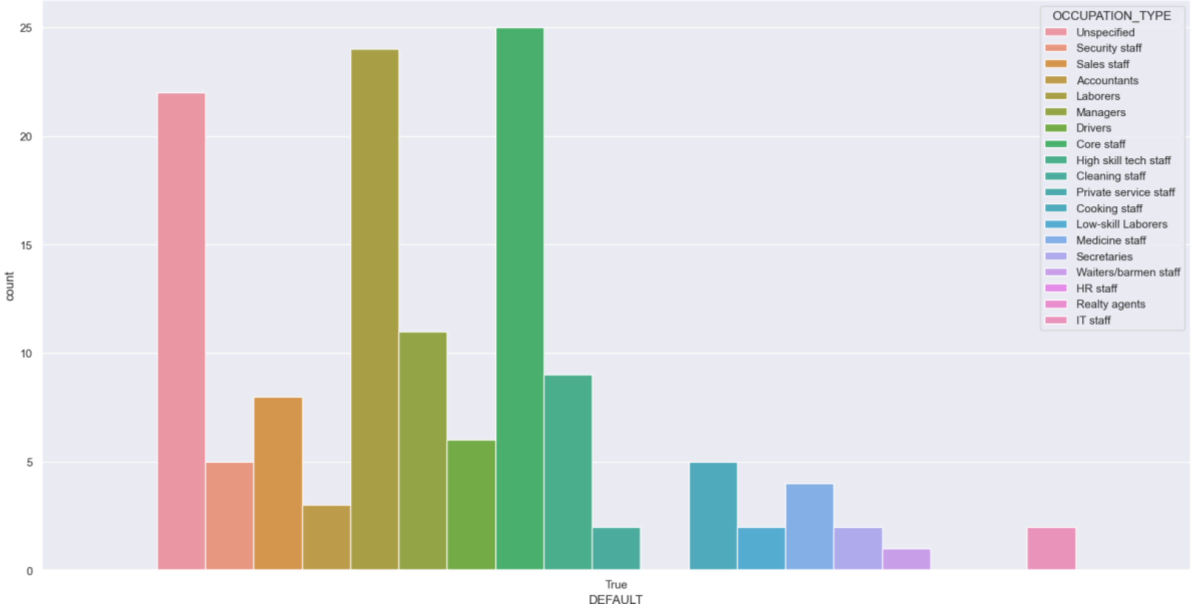
Out[34]:

```
<AxesSubplot:xlabel='DEFAULT', ylabel='count'>
```

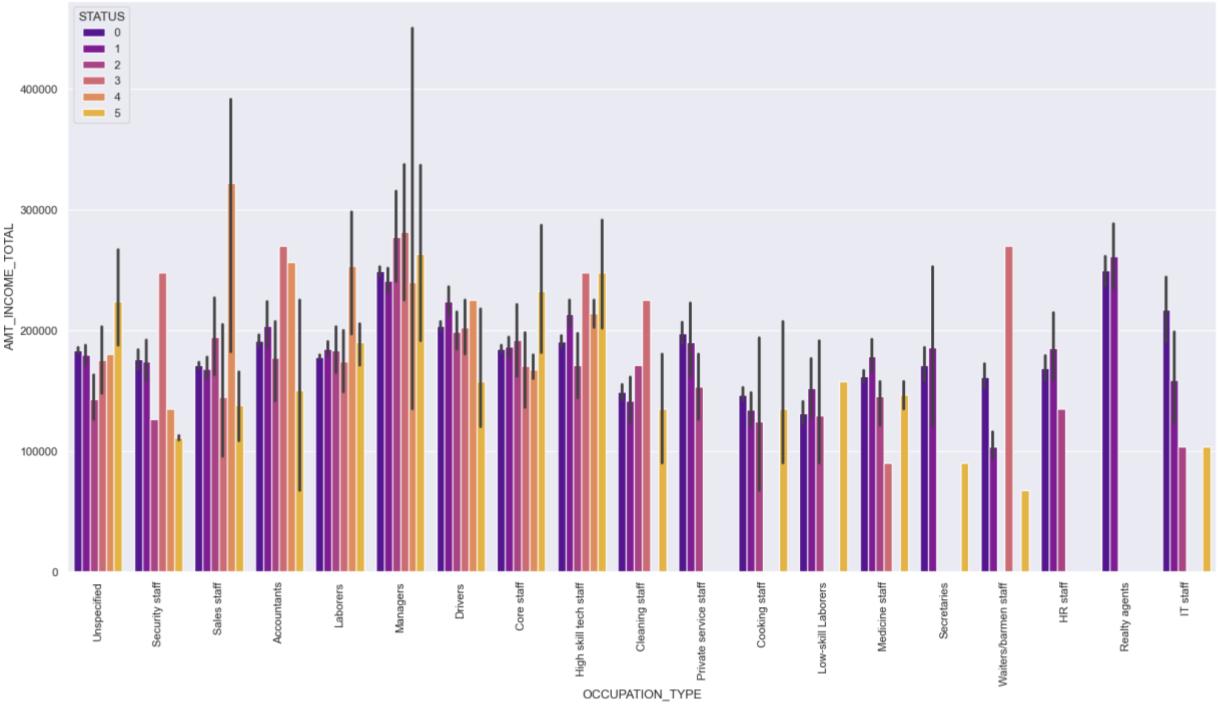


```
In [35]: sns.countplot(data=Target1, x="DEFAULT", hue=application_df['OCCUPATION_T
```

```
Out[35]: <AxesSubplot:xlabel='DEFAULT', ylabel='count'>
```

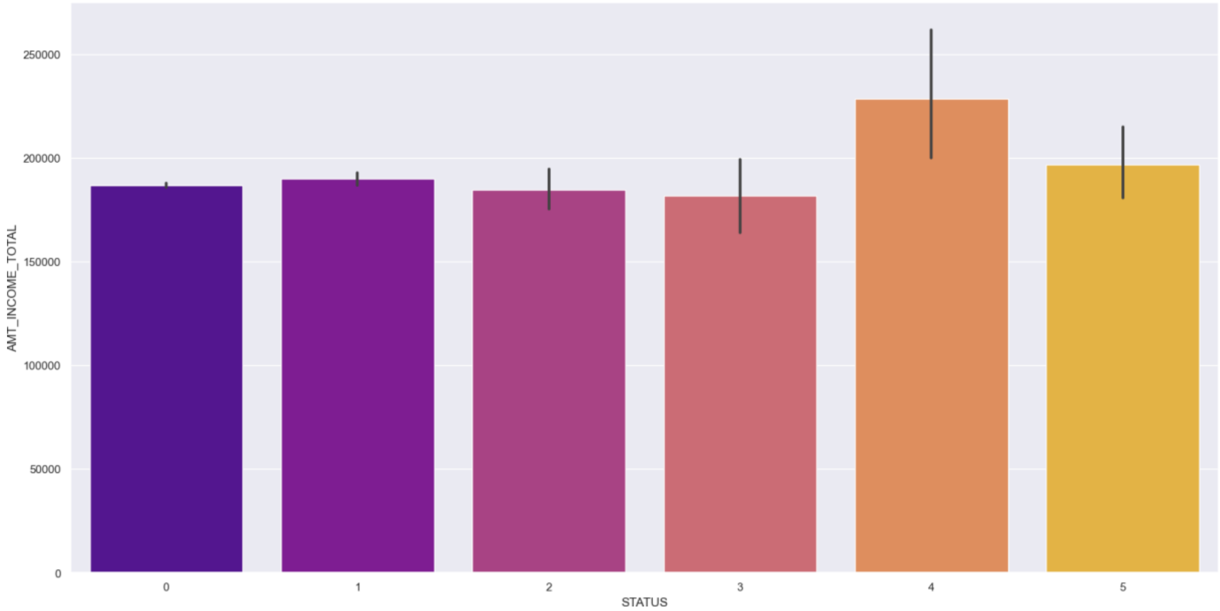


In [36]:
```
import seaborn as sns

import matplotlib.pyplot as plt



sns.set_style('darkgrid')

ax = sns.barplot(x='OCCUPATION_TYPE', y='AMT_INCOME_TOTAL', hue='STATUS',

ax.set_xticklabels(ax.get_xticklabels(), rotation=90)

plt.show()
```



In [37]:
```
import seaborn as sns

import numpy as np

import pandas as pd

# set the background style of the plot

sns.set_style('darkgrid')
sns.barplot(x ='STATUS', y ='AMT_INCOME_TOTAL', data = application_df, es
```

Out[37]:
```
<AxesSubplot:xlabel='STATUS', ylabel='AMT_INCOME_TOTAL'>
```



In [38]:
```
application_df.describe()
```

Out[38]:

	FLAG_OWN_REALTY	AMT_INCOME_TOTAL	AGE	EXP_YEARS	CNT_FA
count	2G775.000000	2G775.000000	2G775.000000	2G775.000000	2
mean	0.348833	1872G2.157G47	3U.U85GUG	G.U42185	
std	0.47GG10	82250.548701	U.4GG401	5.582070	
min	0.000000	27000.000000	20.000000	1.000000	
25%	0.000000	135000.000000	32.000000	3.000000	
50%	0.000000	175500.000000	3U.000000	5.000000	
75%	1.000000	225000.000000	47.000000	U.000000	
max	1.000000	450000.000000	G7.000000	27.000000	

PRE PROCESSING

In [39]:
```
application_df = application_df.drop(['AGE','DEFAULT'],axis=1)
```

In [40]:
```
# Import necessary libraries
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score

# Assign target variable
y = application_df['STATUS']
x = application_df.drop(['STATUS'], axis=1)
```

In [41]:
```
categories = ['NAME_INCOME_TYPE','NAME_EDUCATION_TYPE','NAME_FAMILY_STATU
```

In [42]:

```
# Split the data into training, validation, and testing sets
x_train_val, x_test, y_train_val, y_test = train_test_split(x, y, test_si

# Split the training and validation sets
x_train, x_val, y_train, y_val = train_test_split(x_train_val, y_train_va
```

In [43]:

```
# Numerical and categorical features are handled differently, therefore w

# we get the categorical features by removing the deposit from the list o
cat_feature_names = [item for item in categories if not ( item=='STATUS'
```

In [44]:

```
x.columns
```

Out[44]:

```
Index(['FLAG_OWN_REALTY', 'AMT_INCOME_TOTAL', 'NAME_INCOME_TYPE',
       'NAME_EDUCATION_TYPE', 'NAME_FAMILY_STATUS', 'NAME_HOUSING_TYPE',
       'EXP_YEARS', 'OCCUPATION_TYPE', 'CNT_FAM_MEMBERS'],
      dtype='object')
```

In [45]:

```
x_num = x.drop(cat_feature_names, axis=1)
numerical_feature_names = x_num.columns

x_cat = x[cat_feature_names]
```

In [46]:

```
numerical_feature_names
```

Out[46]:

```
Index(['AMT_INCOME_TOTAL', 'EXP_YEARS', 'CNT_FAM_MEMBERS'], dtype='objec
t')
```

In [47]:

```
# # Categorical features can further be divided in Ordinal and OneHotEnco
# # The only feature suitable for an ordinal encoder is 'education'.
# # For good results, we need to manually the desired order of the classe
ordinal_feature_names = ['NAME_EDUCATION_TYPE']
ordinal_feature_classes = [['Lower secondary', 'Secondary / secondary spe

one_hot_feature_names = [item for item in cat_feature_names if not item i
```

In [48]:

```
class DataFrameSelector(BaseEstimator, TransformerMixin):
    def __init__(self, feature_names):
        self.attribute_names = feature_names
    def fit(self, X, y=None):
        return self
    def transform(self, X):
        return X[self.attribute_names].values
```

In [49]:

```
# One-Hot-Encoder replaces categorical features by boolean features, stat
one_hot_pipeline = Pipeline([
        ('selector', DataFrameSelector(categories)),
        ('one_hot_encoder', OneHotEncoder(drop='first', sparse=False))
    ])

# We run the pipeline to check, whether it runs with no problems
temp = one_hot_pipeline.fit_transform(x)
```

In [50]:
```
# OrdinalEncoder replaced categorical features by their position in a lis
ordinal_pipeline = Pipeline([
        ('selector', DataFrameSelector(ordinal_feature_names)),
        ('ordinal_encoder', OrdinalEncoder(categories=ordinal_feature_cla
    ])
```

In [51]:
```
from sklearn.preprocessing import StandardScaler
num_pipeline = Pipeline([
        ('selector', DataFrameSelector(numerical_feature_names)),
        ('std_scaler', StandardScaler())
    ])
```

In [52]:
```
full_pipeline = FeatureUnion(transformer_list=[
        ("num_pipeline", num_pipeline),
        ("one_hot_pipeline", one_hot_pipeline),
        ("ordinal_pipeline", ordinal_pipeline),
    ])
```

In [53]:
```
df_prepared_train = full_pipeline.fit_transform(x_train)
df_prepared_val = full_pipeline.fit_transform(x_val)
df_prepared_test = full_pipeline.fit_transform(x_test)
```

In [54]:
```
df_prepared_train
```

Out[54]:
```
array([[-1.46234442, -0.88709672, -0.30842635, ...,  0.        ,
         1.        ,  1.        ],
       [ 0.44397579, -0.17068557,  0.79732445, ...,  0.        ,
         0.        ,  1.        ],
       [ 0.44397579, -1.0661995 , -0.30842635, ...,  0.        ,
         1.        ,  1.        ],
       ...,
       [-0.9176815 , -0.17068557,  0.79732445, ...,  0.        ,
         0.        ,  3.        ],
       [ 0.98863871,  3.23226738, -0.30842635, ...,  0.        ,
         1.        ,  1.        ],
       [ 0.44397579,  0.00841722,  0.79732445, ...,  0.        ,
         0.        ,  3.        ]])
```

In [55]:
```
import imblearn
from imblearn.over_sampling import SMOTE

smote = SMOTE(random_state=42)

prepared_smote, label_smote = smote.fit_resample(df_prepared_train,y_trai
#prepared_smote_val, label_smote_val = smote.fit_resample(df_prepared_val
```

In [56]:
```
# get unique values and their counts
unique_values, counts = np.unique(label_smote, return_counts=True)

# print the results
for value, count in zip(unique_values, counts):
    print(f"{value}: {count}")
```

```
0: 14145
1: 14145
2: 14145
3: 14145
4: 14145
5: 14145
```

In [57]:

```
df_prepared_train.shape
```

Out[57]:

```
(16065, 40)
```

In [58]:

```
y_train.shape
```

Out[58]:

```
(16065,)
```

In [59]:

```
prepared_smote.shape
```

Out[59]:

```
(84870, 40)
```

In [60]:

```
label_smote.shape
```

Out[60]:

```
(84870,)
```

In [61]:

```
label_smote
```

Out[61]:

```
0        0
1        0
2        0
3        0
4        1
        ..
84865    5
84866    5
84867    5
84868    5
84869    5
Name: STATUS, Length: 84870, dtype: int64
```

MODELLING

In [62]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import precision_score,accuracy_score,recall_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
import pandas as pd
import numpy as np
from sklearn.metrics import accuracy_score
from imblearn.over_sampling import RandomOverSampler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,classification_report
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import f1_score
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import KFold
from imblearn.over_sampling import RandomOverSampler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,classification_report,confusio
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import plot_confusion_matrix
from sklearn.model_selection import RandomizedSearchCV
import xgboost as xgb
from scipy.stats import randint
```

Model 1- Random Forest

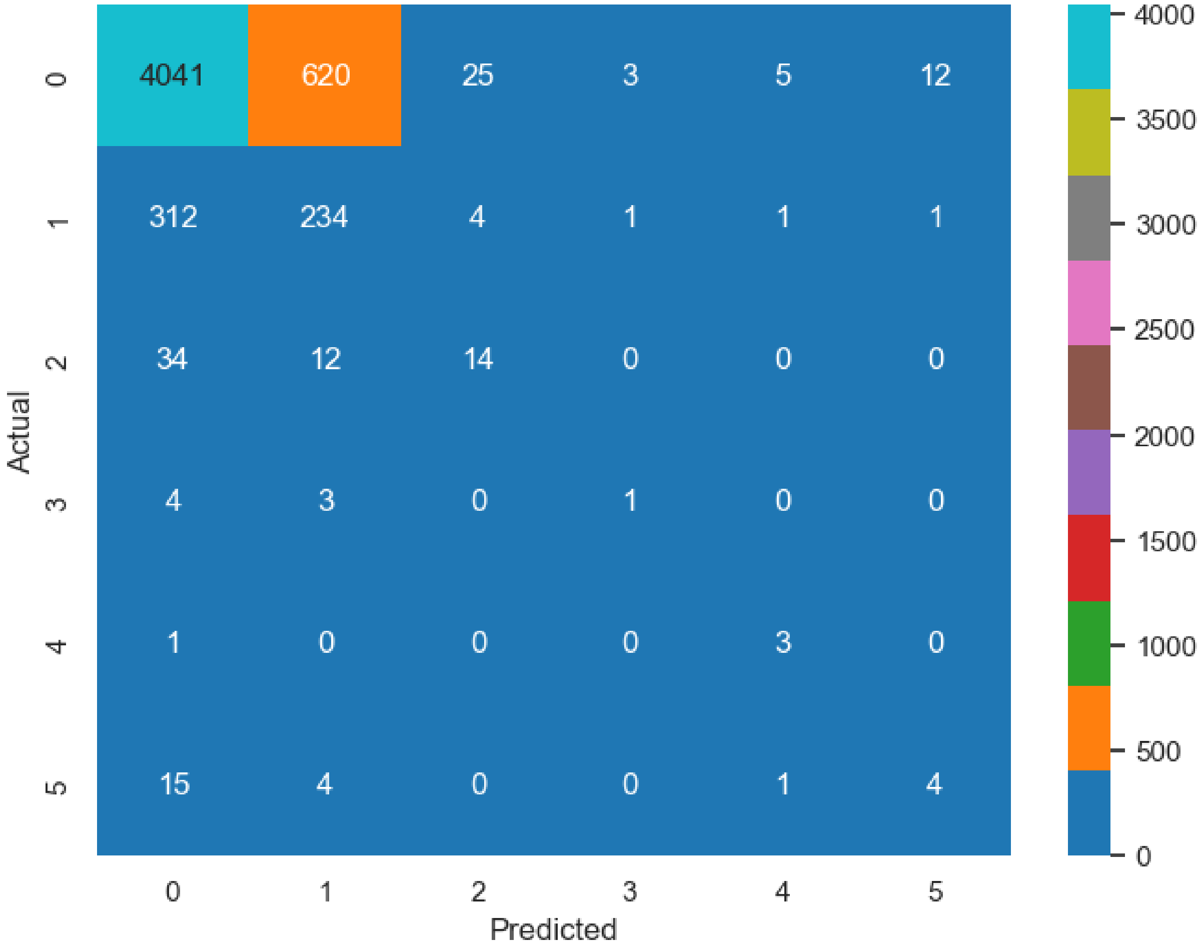
In [63]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Create an instance of the Random Forest classifier with hyperparameters
rf_clf = RandomForestClassifier(n_estimators=200, max_depth=50, random_st

# Fit the model to the training data
rf_clf.fit(prepared_smote,label_smote)

#######

# Evaluate the model on the validation set
y_val_pred = rf_clf.predict(df_prepared_val)
print('Validation set accuracy:', accuracy_score(y_val, y_val_pred))
print(confusion_matrix(y_val, y_val_pred))

cm=confusion_matrix(y_val, y_val_pred)
```

```
Validation set accuracy: 0.8024276377217554
[[4041  620   25    3    5   12]
 [ 312  234    4    1    1    1]
 [  34   12   14    0    0    0]
 [   4    3    0    1    0    0]
 [   1    0    0    0    3    0]
 [  15    4    0    0    1    4]]
```

In [64]:
```
import matplotlib.pyplot as plt
import seaborn as sns
classes=['0','1','2','3','4','5']
def plot_confusion_matrix(cm, classes):
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt='g', cmap='tab10', xticklabels=classe
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.show()
```

In [65]:
```
plot_confusion_matrix(cm, classes)
```

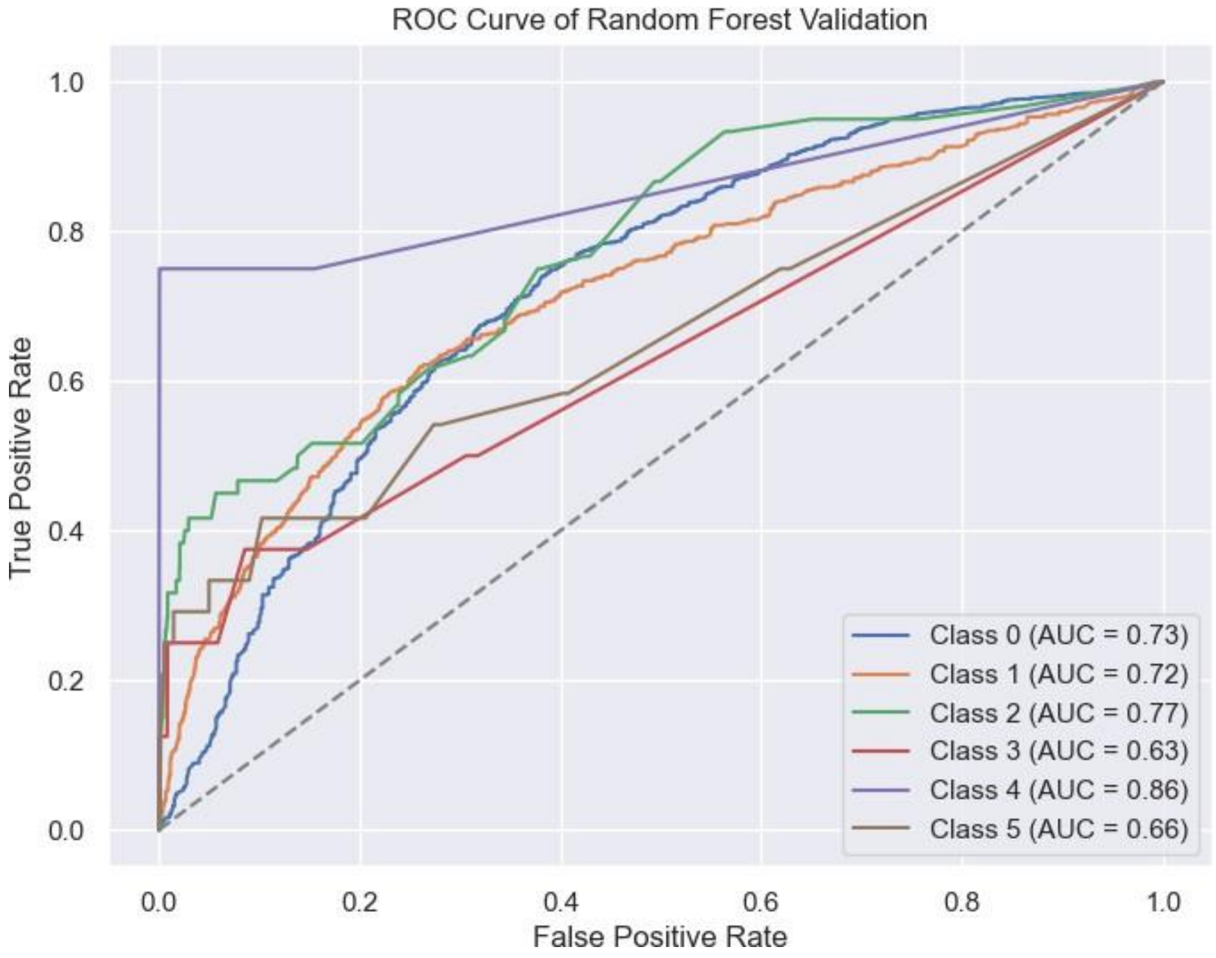


In [66]:
```
print(classification_report(y_val, y_val_pred))
```

```
              precision    recall  f1-score   support

           0       0.92      0.86      0.89      4706
           1       0.27      0.42      0.33       553
           2       0.33      0.23      0.27        60
           3       0.20      0.12      0.15         8
           4       0.30      0.75      0.43         4
           5       0.24      0.17      0.20        24

    accuracy                           0.80      5355
   macro avg       0.37      0.43      0.38      5355
weighted avg       0.84      0.80      0.82      5355
```

In [67]:

```
import matplotlib.pyplot as plt

# Get the predicted probability of each class
y_proba = rf_clf.predict_proba(df_prepared_val)

# Compute the ROC curve and ROC AUC score for each class
fpr  = dict()
tpr = dict()
roc_auc = dict()
for i in range(6):
    fpr[i], tpr[i], _ = roc_curve(y_val == i, y_proba[:, i])
    roc_auc[i] = roc_auc_score(y_val == i, y_proba[:, i])

# Plot the ROC curve for each class
plt.figure(figsize=(8,6))
for i in range(6):
    plt.plot(fpr[i], tpr[i], label='Class {} (AUC = {:.2f})'.format(i, ro
plt.plot([0, 1], [0, 1], linestyle='--', color='grey')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve of Random Forest Validation')
plt.legend()
plt.show()
```



MODEL 2 - KNN

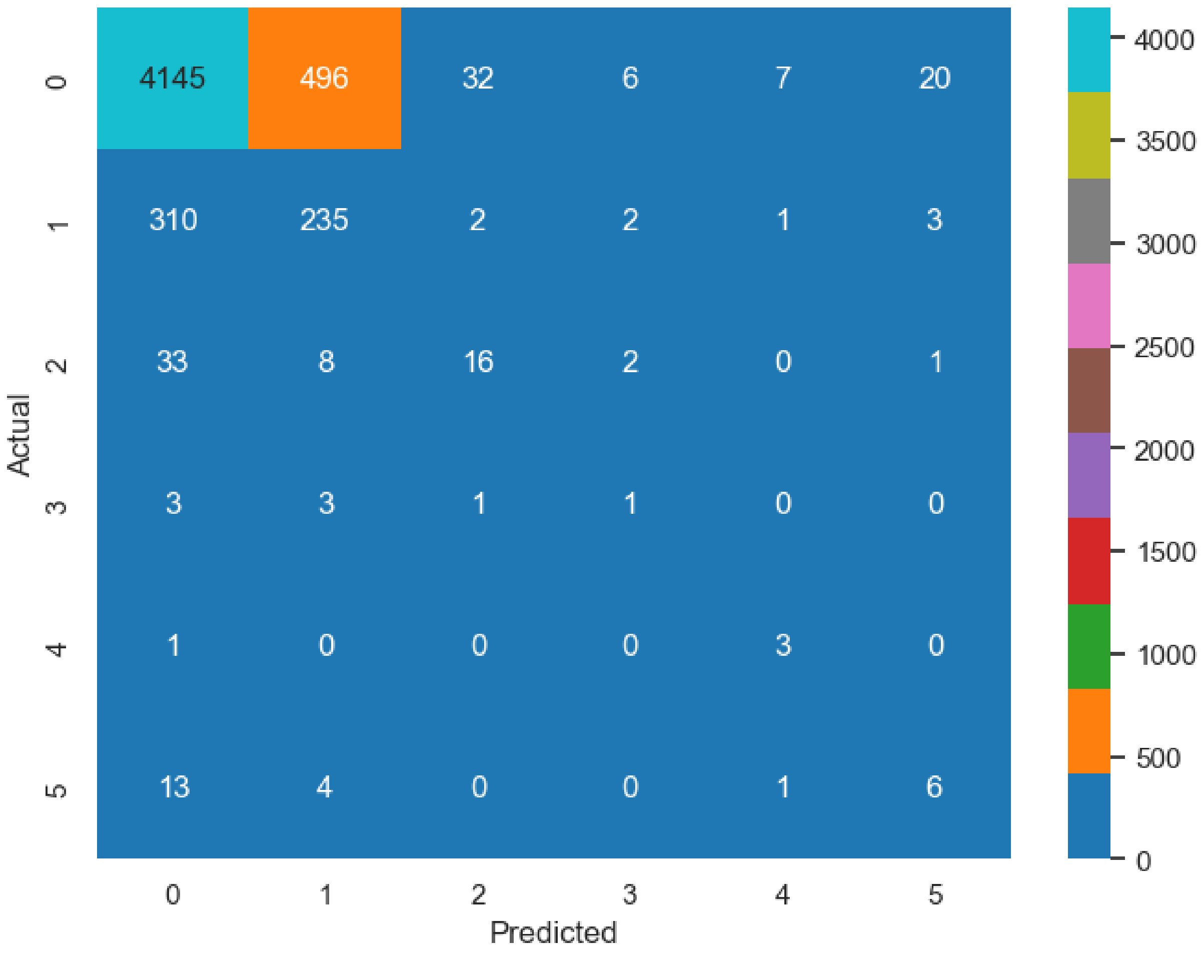
In [68]:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report



# Create an instance of the KNeighborsClassifier with k=5
knn_clf = KNeighborsClassifier(n_neighbors=5,weights='distance',p=1)

# Fit the model to the training data
knn_clf.fit(prepared_smote,label_smote)

# Evaluate the model on the validation set
y_val_pred = knn_clf.predict(df_prepared_val)
print('Validation set accuracy:', accuracy_score(y_val, y_val_pred))
print(confusion_matrix(y_val, y_val_pred))
print(classification_report(y_val, y_val_pred))
cm=confusion_matrix(y_val, y_val_pred)
```

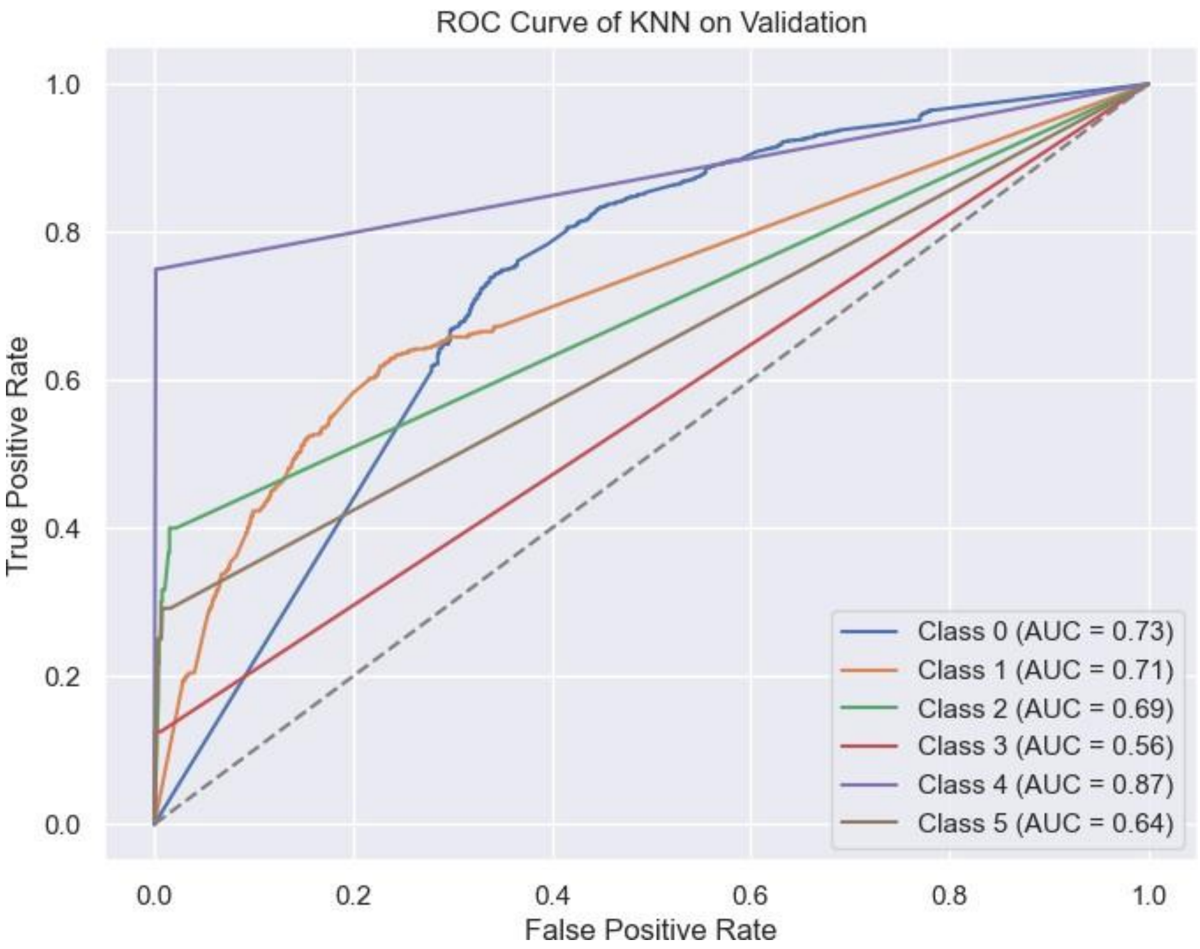
```
Validation set accuracy: 0.822782446311858
[[4145  496   32    6    7   20]
 [ 310  235    2    2    1    3]
 [  33    8   16    2    0    1]
 [   3    3    1    1    0    0]
 [   1    0    0    0    3    0]
 [  13    4    0    0    1    6]]
              precision    recall  f1-score   support

           0       0.92      0.88      0.90      4706
           1       0.32      0.42      0.36       553
           2       0.31      0.27      0.29        60
           3       0.09      0.12      0.11         8
           4       0.25      0.75      0.38         4
           5       0.20      0.25      0.22        24

    accuracy                           0.82      5355
   macro avg       0.35      0.45      0.38      5355
weighted avg       0.85      0.82      0.83      5355
```

In [69]:

```
plot_confusion_matrix(cm, classes)
```



In [70]:
```
import matplotlib.pyplot as plt

# Get the predicted probability of each class
y_proba = knn_clf.predict_proba(df_prepared_val)

# Compute the ROC curve and ROC AUC score for each class
fpr  = dict()
tpr = dict()
roc_auc = dict()
for i in range(6):
    fpr[i], tpr[i], _ = roc_curve(y_val == i, y_proba[:, i])
    roc_auc[i] = roc_auc_score(y_val == i, y_proba[:, i])

# Plot the ROC curve for each class
plt.figure(figsize=(8,6))
for i in range(6):
    plt.plot(fpr[i], tpr[i], label='Class {} (AUC = {:.2f})'.format(i, ro
plt.plot([0, 1], [0, 1], linestyle='--', color='grey')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve of KNN on Validation')
plt.legend()
plt.show()
```

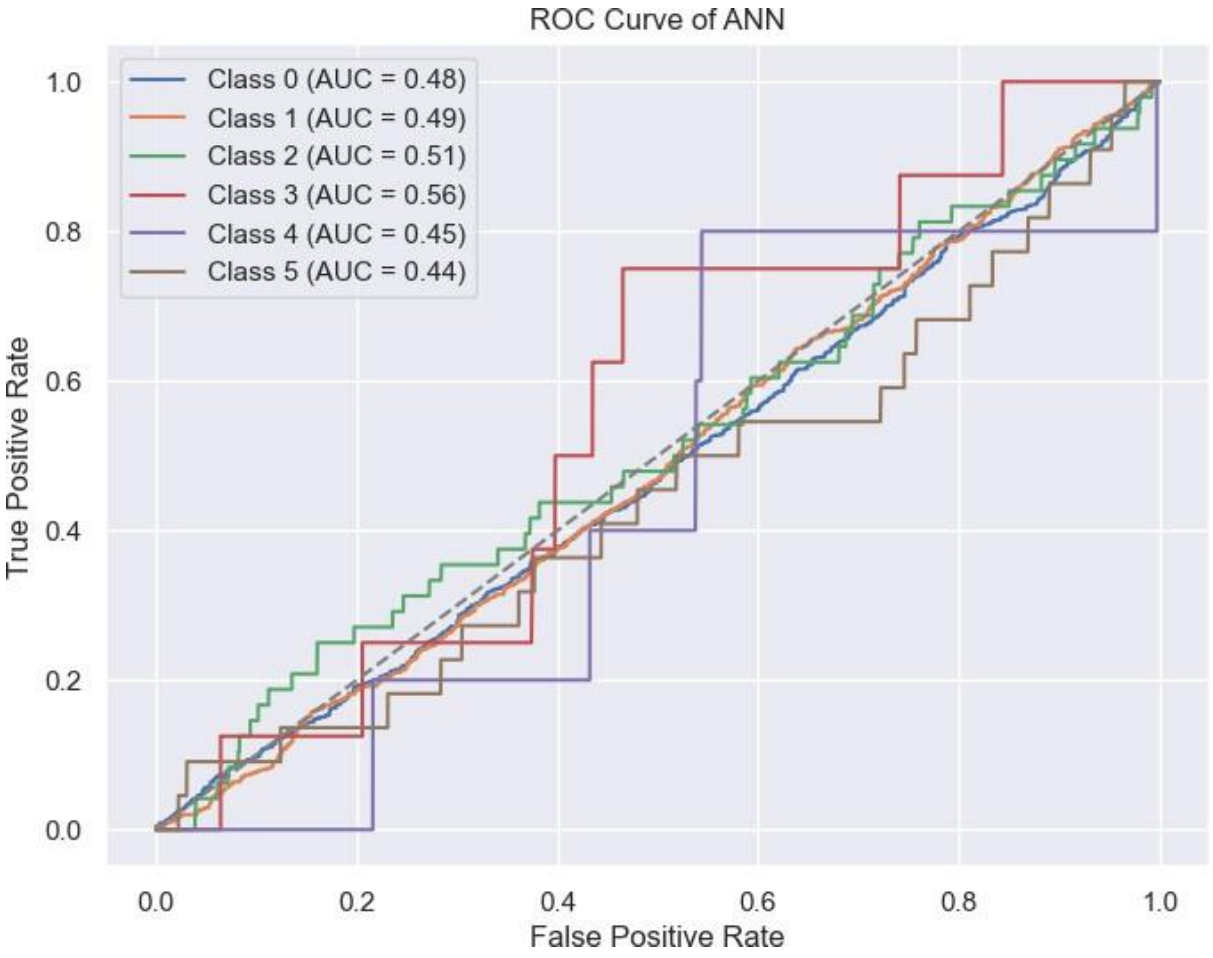


Model 3- ANN

In [71]:

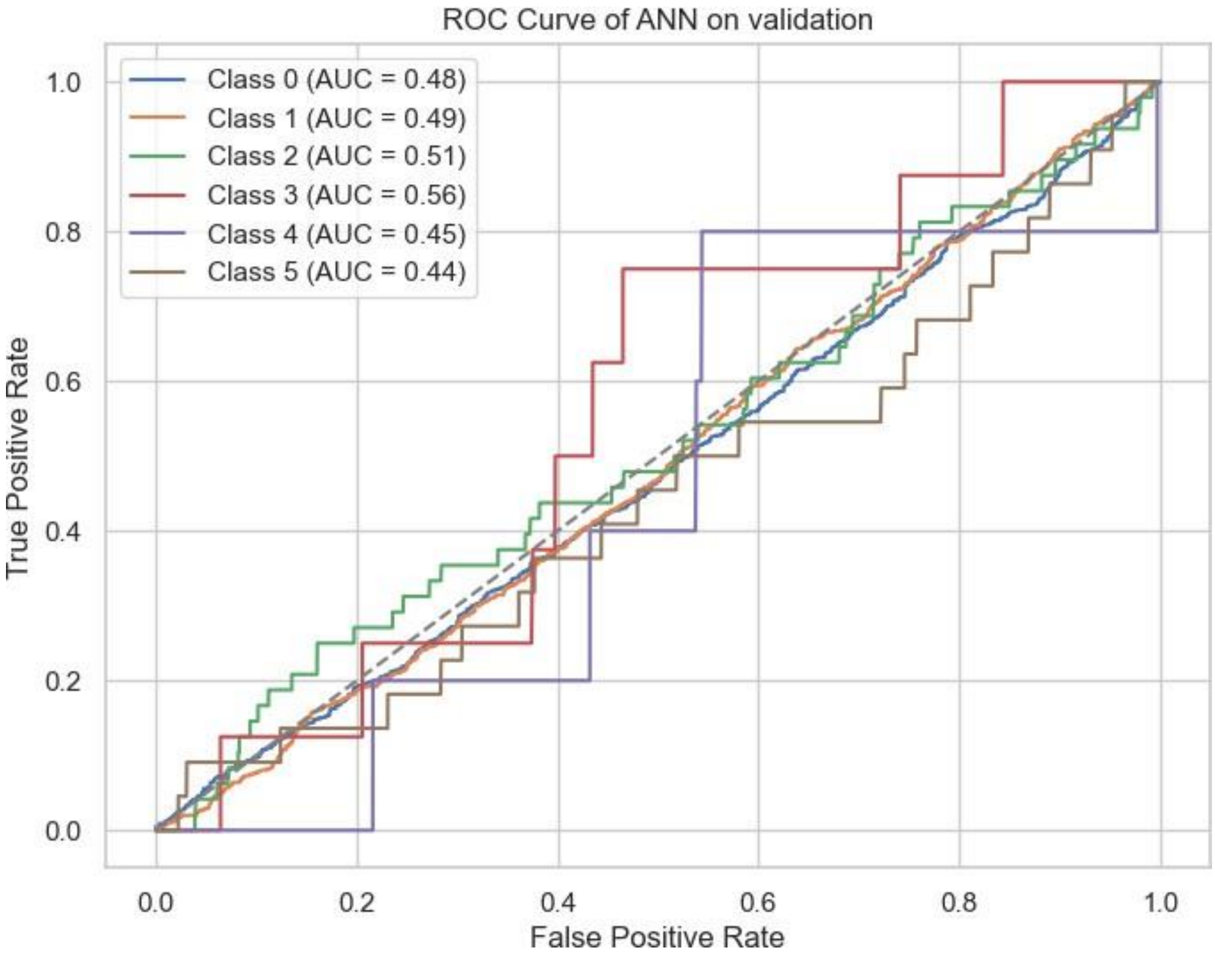
```
# Create an instance of the MLPClassifier with hyperparameters
mlp_clf = MLPClassifier(hidden_layer_sizes=(100,), activation='relu', alp
# Fit the model to the training data
mlp_clf.fit(prepared_smote,label_smote)
# Evaluate the model on the validation set
y_pred_val = mlp_clf.predict(df_prepared_val)
print(confusion_matrix(y_val, y_pred_val))
print(classification_report(y_val, y_pred_val))
accuracy = mlp_clf.score(df_prepared_val, y_val)
print('Validation accuracy:', accuracy)
import matplotlib.pyplot as plt
```

```
[[3435 1069   96   30   13   63]
 [ 256  272   11    3    4    7]
 [  24    8   22    1    0    5]
 [   3    3    0    2    0    0]
 [   0    1    0    0    3    0]
 [  12    4    0    0    1    7]]
              precision    recall  f1-score   support

           0       0.92      0.73      0.81      4706
           1       0.20      0.49      0.28       553
           2       0.17      0.37      0.23        60
           3       0.06      0.25      0.09         8
           4       0.14      0.75      0.24         4
           5       0.09      0.29      0.13        24

    accuracy                           0.70      5355
   macro avg       0.26      0.48      0.30      5355
weighted avg       0.83      0.70      0.75      5355

Validation accuracy: 0.6985994397759103
```



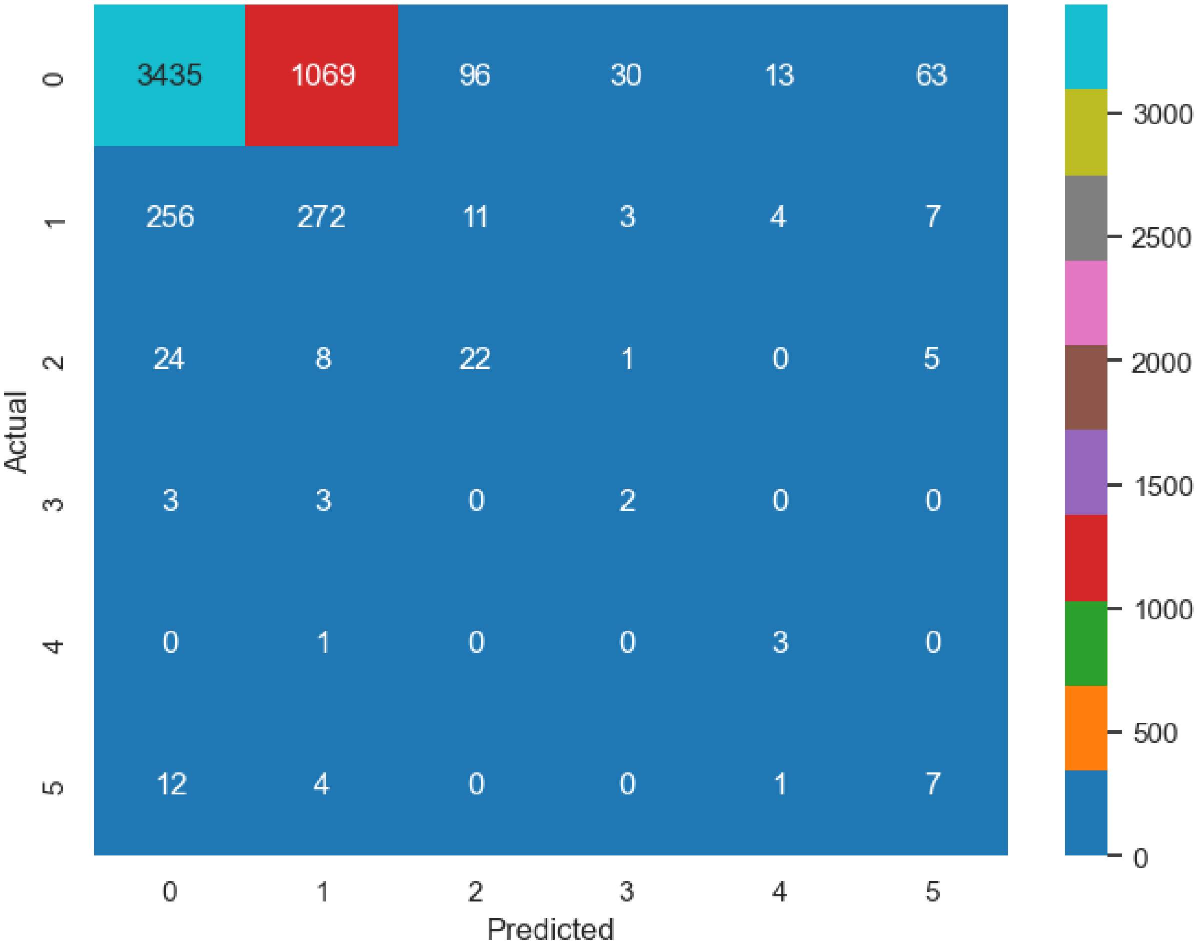
In [105…

```
# Get the predicted probability of each class
y_proba = mlp_clf.predict_proba(df_prepared_val)

# Compute the ROC curve and ROC AUC score for each class
fpr  =  dict()
tpr = dict()
roc_auc = dict()
for i in range(6):
    fpr[i], tpr[i], _ = roc_curve(y_test == i, y_proba[:, i])
    roc_auc[i] = roc_auc_score(y_test == i, y_proba[:, i])


# Plot the ROC curve for each class
plt.figure(figsize=(8,6))
for i in range(6):
    plt.plot(fpr[i], tpr[i], label='Class {} (AUC = {:.2f})'.format(i, ro
plt.plot([0, 1], [0, 1], linestyle='--', color='grey')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve of ANN on validation')
plt.legend()
plt.show()
```



In [73]:

```
cm=confusion_matrix(y_val, y_pred_val)
plot_confusion_matrix(cm, classes)
```

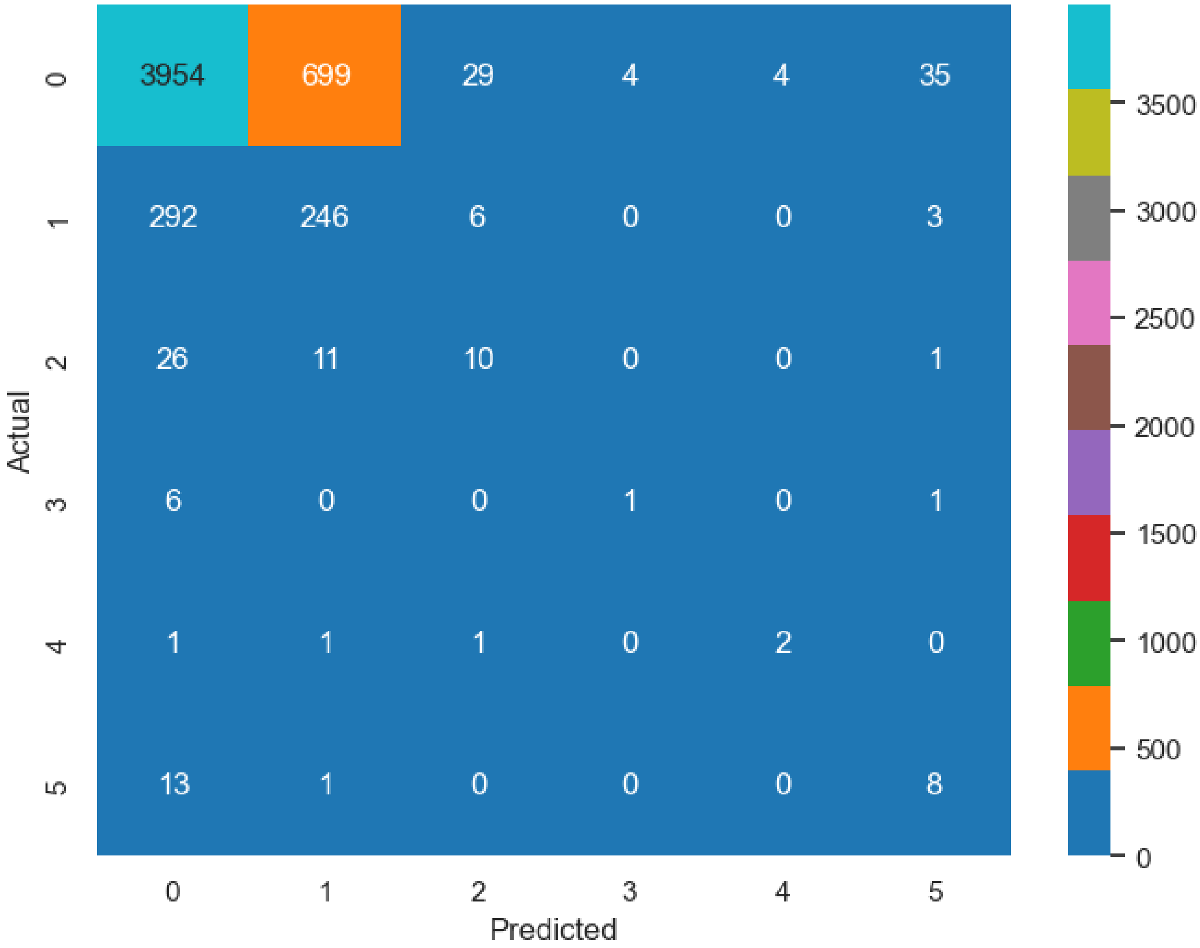


Test Set on Random Forest based on precision and recall of validation

In [74]:
```
# Evaluate the model on the test set
y_test_pred = rf_clf.predict(df_prepared_test)
print('Test set accuracy:', accuracy_score(y_test, y_test_pred))
print(confusion_matrix(y_test, y_test_pred))
print(classification_report(y_test, y_test_pred))
cm=confusion_matrix(y_test, y_test_pred)
```

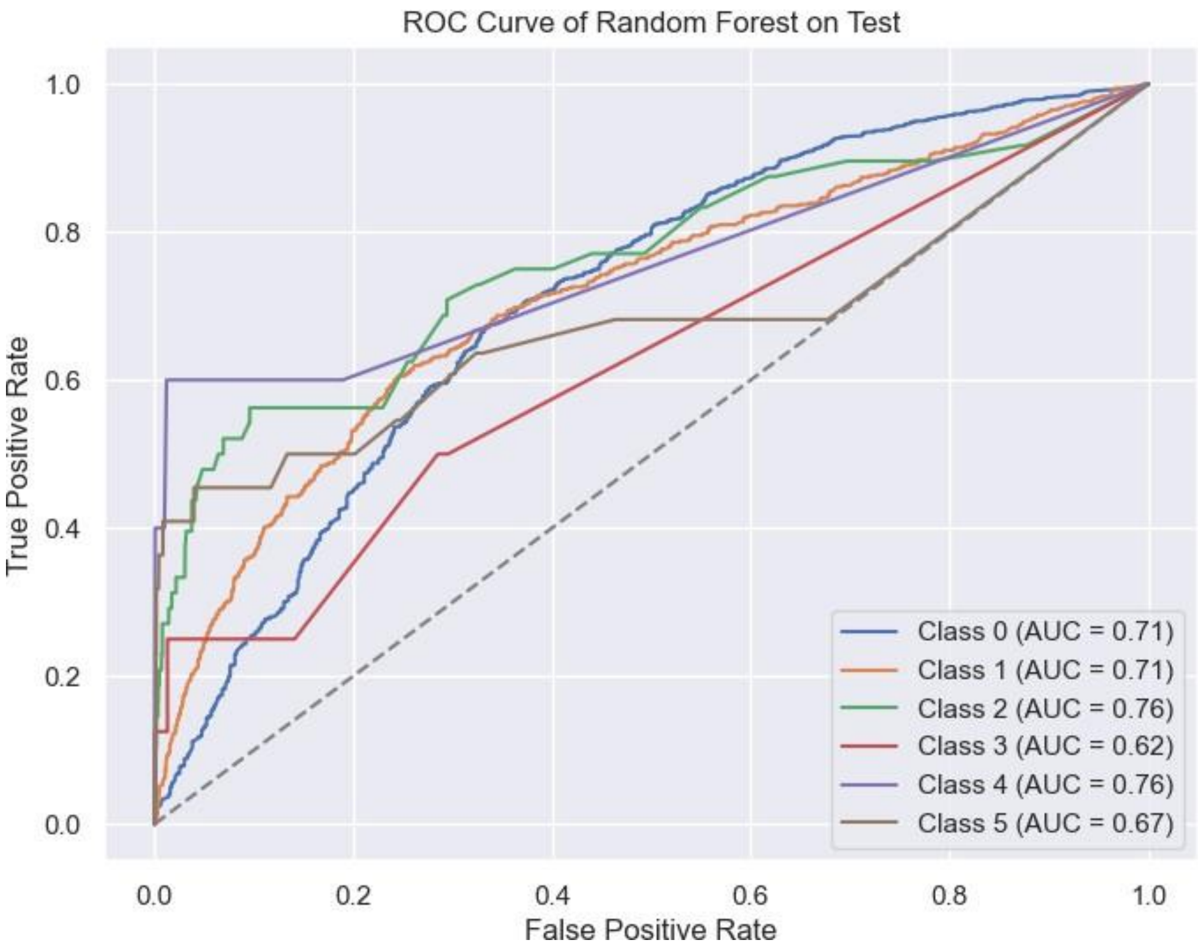
```
Test set accuracy: 0.788235294117647
[[3954  699   29    4    4   35]
 [ 292  246    6    0    0    3]
 [  26   11   10    0    0    1]
 [   6    0    0    1    0    1]
 [   1    1    1    0    2    0]
 [  13    1    0    0    0    8]]
              precision    recall  f1-score   support

           0       0.92      0.84      0.88      4725
           1       0.26      0.45      0.33       547
           2       0.22      0.21      0.21        48
           3       0.20      0.12      0.15         8
           4       0.33      0.40      0.36         5
           5       0.17      0.36      0.23        22

    accuracy                           0.79      5355
   macro avg       0.35      0.40      0.36      5355
weighted avg       0.84      0.79      0.81      5355
```

In [75]:
```
plot_confusion_matrix(cm, classes)
```



In [76]:

```
import matplotlib.pyplot as plt

# Get the predicted probability of each class
y_proba = rf_clf.predict_proba(df_prepared_test)

# Compute the ROC curve and ROC AUC score for each class
fpr  = dict()
tpr = dict()
roc_auc = dict()
for i in range(6):
    fpr[i], tpr[i], _ = roc_curve(y_test == i, y_proba[:, i])
    roc_auc[i] = roc_auc_score(y_test == i, y_proba[:, i])

# Plot the ROC curve for each class
plt.figure(figsize=(8,6))
for i in range(6):
    plt.plot(fpr[i], tpr[i], label='Class {} (AUC = {:.2f})'.format(i, ro
plt.plot([0, 1], [0, 1], linestyle='--', color='grey')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve of Random Forest on Test')
plt.legend()
plt.show()
```



Extracting Predictions to Dataframe

In [77]:
```
feature_names = x_test.columns.tolist()
```

In [78]:
```
print(feature_names)
```

```
['FLAG_OWN_REALTY', 'AMT_INCOME_TOTAL', 'NAME_INCOME_TYPE', 'NAME_EDUCAT
ION_TYPE', 'NAME_FAMILY_STATUS', 'NAME_HOUSING_TYPE', 'EXP_YEARS', 'OCCU
PATION_TYPE', 'CNT_FAM_MEMBERS']
```

In [79]:
```
from sklearn.preprocessing import StandardScaler
predictions=y_test_pred
features=x_test
type(x_test)
```

Out[79]:
```
pandas.core.frame.DataFrame
```

In [80]:
```
x_test['STATUS']=predictions

df=x_test
```

In [81]:
```
x_test.describe()
```

Out[81]:

	FLAG_OWN_REALTY	AMT_INCOME_TOTAL	EXP_YEARS	CNT_FAM_MEMBERS	
count	5355.000000	5355.000000	5355.000000	5355.000000	53
mean	0.348G4G	185222.3G80G7	G.U215GU	2.2U747U	
std	0.47G58G	81354.0703G5	5.557U11	0.U1750U	
min	0.000000	31500.000000	1.000000	1.000000	
25%	0.000000	135000.000000	3.000000	2.000000	
50%	0.000000	1GG500.000000	5.000000	2.000000	
75%	1.000000	225000.000000	U.000000	3.000000	
max	1.000000	450000.000000	27.000000	5.000000	

Business use case

Assigning a Risk Scale Based on Status

0-1-2 = low risk

3-4 = moderate risk

5 = High risk

In [82]:

```
import pandas as pd

def add_risk_degree(df):
    """
    Adds a 'risk_degree' column to the given dataframe based on the value
    """
    # Create a dictionary to map status values to risk degree values
    status_to_risk = {
        0: 'Low Risk',
        1: 'Low Risk',
        2: 'Low Risk',
        3: 'Moderate Risk',
        4: 'Moderate Risk',
        5: 'High Risk'
    }

    # Add a new column to the dataframe using the mapping
    df['Risk_Degree'] = df['STATUS'].map(status_to_risk)

    return df
```

In [83]:

```
df=add_risk_degree(df)
```

In [84]:

```
df.head()
```

Out[84]:

	FLAG_OWN_REALTY	AMT_INCOME_TOTAL	NAME_INCOME_TYPE	NAME_EDUCAT
11008	0	247500.0	State servant	Higher
227C4	0	12G000.0	Commercial associate	Secondary /
25434	1	225000.0	Commercial associate	Secondary /
30988	1	112500.0	State servant	Secondary /
10891	0	247500.0	Working	Secondary /

In [85]:

```
import pandas as pd

# Define the function to calculate credit limit
def calculate_credit_limit(df, dti_ratio):
    monthly_debt_obligation = 1500
    df['CreditLimit'] = (df['AMT_INCOME_TOTAL']/12 * dti_ratio) - monthly
    return df
```

In [86]:

```
df=calculate_credit_limit(df,0.4)
```

In [87]:

```
df.loc[df['STATUS']==5, 'CreditLimit']=0
```

In [88]:

```
df.describe()
```

Out[88]:

	FLAG_OWN_REALTY	AMT_INCOME_TOTAL	EXP_YEARS	CNT_FAM_MEMBERS	
count	5355.000000	5355.000000	5355.000000	5355.000000	53
mean	0.348G4G	185222.3G80G7	G.U215GU	2.2U747U	
std	0.47G58G	81354.0703G5	5.557U11	0.U1750U	
min	0.000000	31500.000000	1.000000	1.000000	
25%	0.000000	135000.000000	3.000000	2.000000	
50%	0.000000	1GG500.000000	5.000000	2.000000	
75%	1.000000	225000.000000	U.000000	3.000000	
max	1.000000	450000.000000	27.000000	5.000000	

In [89]:

```
#Profit Per Customer

# Define the interest rate
interest_rate = 0.22

# Calculate the profit for each row in the DataFrame
df['Profit'] = df['CreditLimit'] * interest_rate * df['STATUS']

# Print the DataFrame with the Profit column added
print(df.head())
```

```
       FLAG_OWN_REALTY  AMT_INCOME_TOTAL       NAME_INCOME_TYPE  \
11008                0          247500.0          State servant   
22764                0          126000.0   Commercial associate   
25434                1          225000.0   Commercial associate   
30988                1          112500.0          State servant   
10891                0          247500.0                Working   

                 NAME_EDUCATION_TYPE     NAME_FAMILY_STATUS  NAME_HOUSING
_TYPE  \
11008               Higher education   Single / not married  House / apar
tment   
22764  Secondary / secondary special                Married  House / apar
tment   
25434  Secondary / secondary special         Civil marriage  House / apar
tment   
30988  Secondary / secondary special   Single / not married  House / apar
tment   
10891  Secondary / secondary special                Married  House / apar
tment   

       EXP_YEARS        OCCUPATION_TYPE   CNT_FAM_MEMBERS  STATUS Risk_De
gree  \
11008          2  High skill tech staff               1.0       0    Low 
Risk   
22764          2             Core staff               2.0       0    Low 
Risk   
25434          1            Accountants               3.0       0    Low 
Risk   
30988         15         Security staff               1.0       1    Low 
Risk   
10891          9               Managers               2.0       0    Low 
Risk   

       CreditLimit  Profit  
11008       6750.0     0.0  
22764       2700.0     0.0  
25434       6000.0     0.0  
30988       2250.0   495.0  
10891       6750.0     0.0  
```

In [90]:

```
df.describe()
```

Out[90]:

	FLAG_OWN_REALTY	AMT_INCOME_TOTAL	EXP_YEARS	CNT_FAM_MEMBERS	
count	5355.000000	5355.000000	5355.000000	5355.000000	53
mean	0.348G4G	185222.3G80G7	G.U215GU	2.2U747U	
std	0.47G58G	81354.0703G5	5.557U11	0.U1750U	
min	0.000000	31500.000000	1.000000	1.000000	
25%	0.000000	135000.000000	3.000000	2.000000	
50%	0.000000	1GG500.000000	5.000000	2.000000	
75%	1.000000	225000.000000	U.000000	3.000000	
max	1.000000	450000.000000	27.000000	5.000000	

For Loss we assume a proportion of customers will fail to pay back their loans

In [91]:

```
#loss Per Customer


# Calculate the profit for each row in the DataFrame
df['Potential Loss'] = df['CreditLimit']

# Print the DataFrame with the Profit column added
print(df.head())
```

```
       FLAG_OWN_REALTY  AMT_INCOME_TOTAL      NAME_INCOME_TYPE  \
11008                0          247500.0         State servant   
22764                0          126000.0  Commercial associate   
25434                1          225000.0  Commercial associate   
30988                1          112500.0         State servant   
10891                0          247500.0               Working   

                 NAME_EDUCATION_TYPE    NAME_FAMILY_STATUS  NAME_HOUSING
_TYPE  \
11008               Higher education  Single / not married  House / apar
tment   
22764  Secondary / secondary special               Married  House / apar
tment   
25434  Secondary / secondary special        Civil marriage  House / apar
tment   
30988  Secondary / secondary special  Single / not married  House / apar
tment   
10891  Secondary / secondary special               Married  House / apar
tment   

       EXP_YEARS        OCCUPATION_TYPE  CNT_FAM_MEMBERS  STATUS Risk_De
gree  \
11008          2  High skill tech staff              1.0       0    Low 
Risk   
22764          2             Core staff              2.0       0    Low 
Risk   
25434          1            Accountants              3.0       0    Low 
Risk   
30988         15         Security staff              1.0       1    Low 
Risk   
10891          9               Managers              2.0       0    Low 
Risk   

       CreditLimit  Profit  Potential Loss  
11008       6750.0     0.0          6750.0  
22764       2700.0     0.0          2700.0  
25434       6000.0     0.0          6000.0  
30988       2250.0   495.0          2250.0  
10891       6750.0     0.0          6750.0  
```

In [92]:

```
df.head()
```

Out[92]:

	FLAG_OWN_REALTY	AMT_INCOME_TOTAL	NAME_INCOME_TYPE	NAME_EDUCAT
11008	0	247500.0	State servant	Higher
227C4	0	12G000.0	Commercial associate	Secondary /
25434	1	225000.0	Commercial associate	Secondary /
30988	1	112500.0	State servant	Secondary /
10891	0	247500.0	Working	Secondary /

In [93]:

```
import pandas as pd
default_rate=0.08
def potential_loss(df,default_rate):
    return df['Potential Loss'].sum()*default_rate
```

In [94]:

```
potential_loss(df,default_rate)
```

Out[94]:

```
1983764.616
```

In [95]:

```
df['Profit'].sum()
```

Out[95]:

```
1091406.2280000001
```

In [96]:

```
df.columns
```

Out[96]:

```
Index(['FLAG_OWN_REALTY', 'AMT_INCOME_TOTAL', 'NAME_INCOME_TYPE',
       'NAME_EDUCATION_TYPE', 'NAME_FAMILY_STATUS', 'NAME_HOUSING_TYPE',
       'EXP_YEARS', 'OCCUPATION_TYPE', 'CNT_FAM_MEMBERS', 'STATUS',
       'Risk_Degree', 'CreditLimit', 'Profit', 'Potential Loss'],
      dtype='object')
```

In [97]:

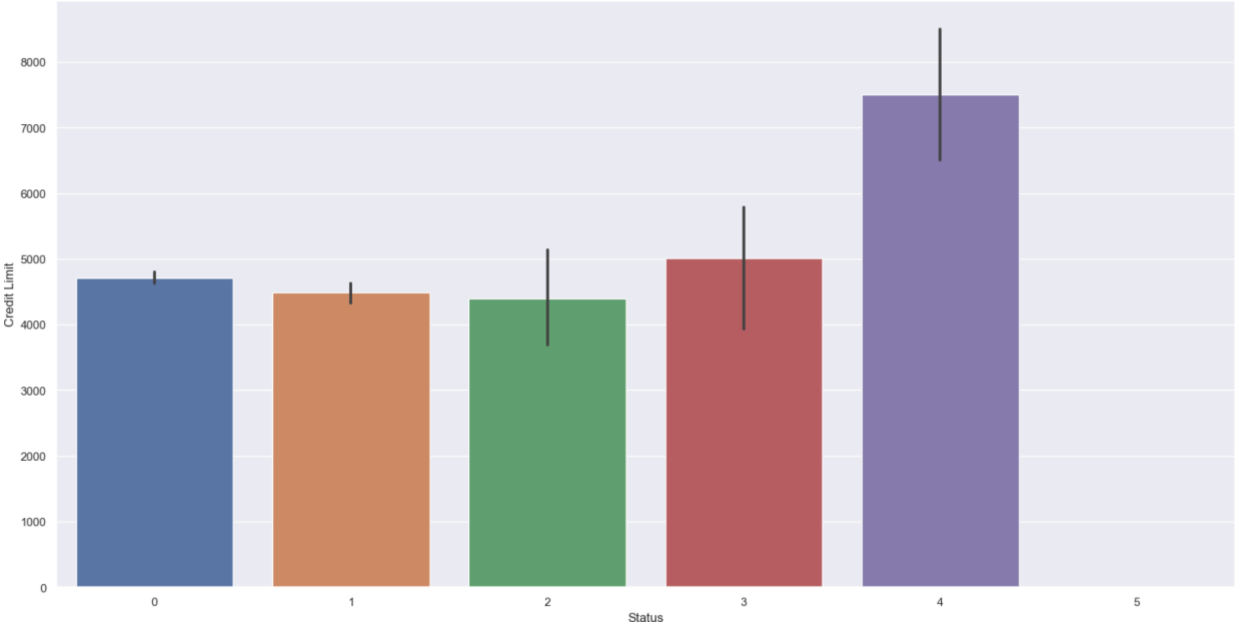
```
df.head()
```

Out[97]:

	FLAG_OWN_REALTY	AMT_INCOME_TOTAL	NAME_INCOME_TYPE	NAME_EDUCAT
11008	0	247500.0	State servant	Higher
227C4	0	12G000.0	Commercial associate	Secondary /
25434	1	225000.0	Commercial associate	Secondary /
30988	1	112500.0	State servant	Secondary /
10891	0	247500.0	Working	Secondary /

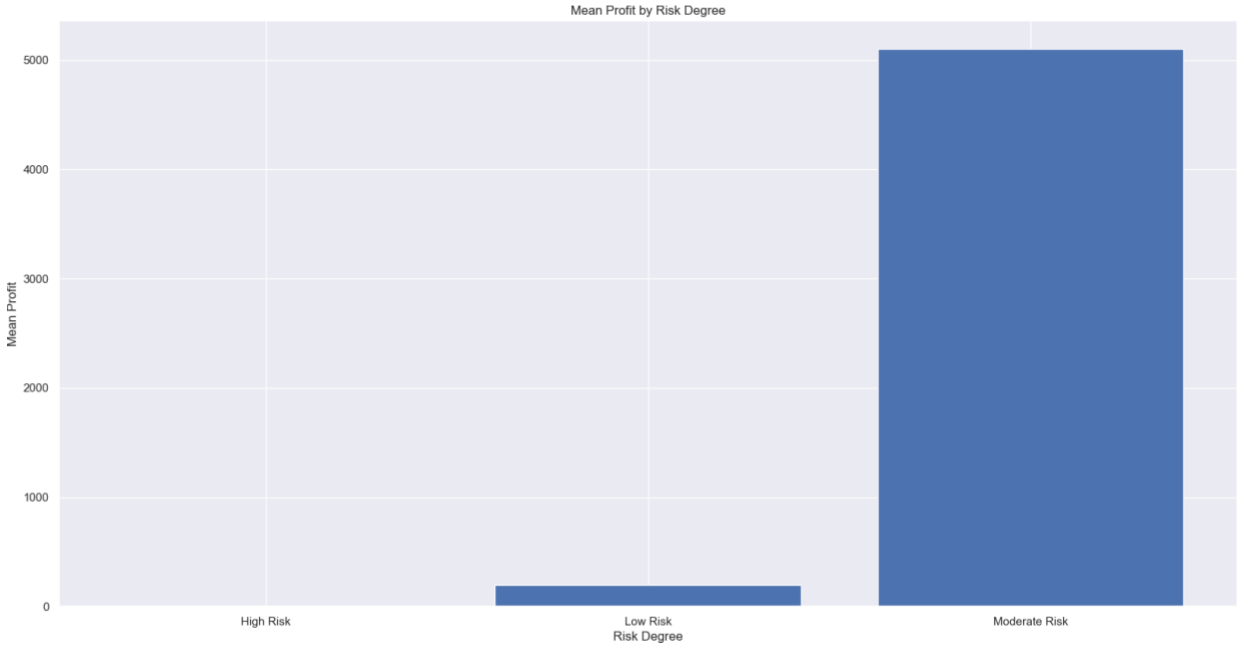
In [98]:

```
import seaborn as sns
import matplotlib.pyplot as plt

# Create scatter plot
sns.barplot(data=df, x='STATUS', y='CreditLimit')

# Set labels
plt.xlabel('Status')
plt.ylabel('Credit Limit')

# Show plot
plt.show()
```



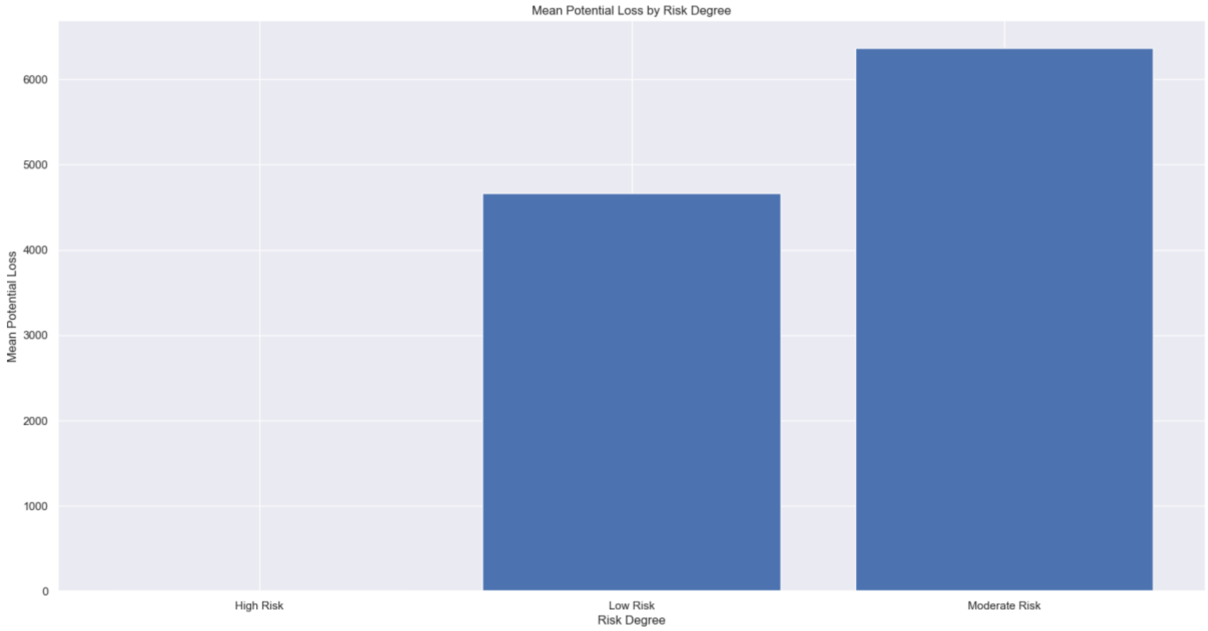
In [99]:

```
import matplotlib.pyplot as plt
import pandas as pd

# Calculate mean profit for each unique Risk_Degree
mean_profit = df.groupby('Risk_Degree')['Profit'].mean()

# Create bar chart
plt.bar(mean_profit.index, mean_profit.values)
plt.xlabel('Risk Degree')
plt.ylabel('Mean Profit')
plt.title('Mean Profit by Risk Degree')
plt.show()
```



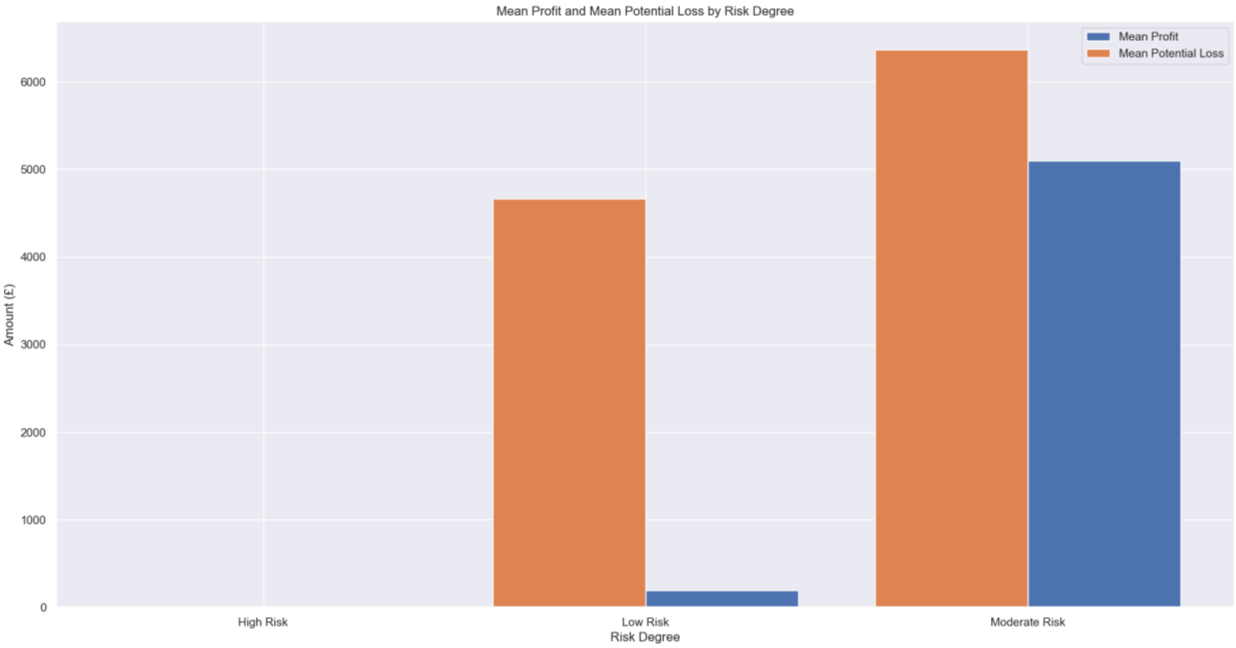
In [100…

```
import matplotlib.pyplot as plt
import pandas as pd

# Calculate mean potential loss for each unique Risk_Degree
mean_potential_loss = df.groupby('Risk_Degree')['Potential Loss'].mean()

# Create bar chart
plt.bar(mean_potential_loss.index, mean_potential_loss.values)
plt.xlabel('Risk Degree')
plt.ylabel('Mean Potential Loss')
plt.title('Mean Potential Loss by Risk Degree')
plt.show()
```



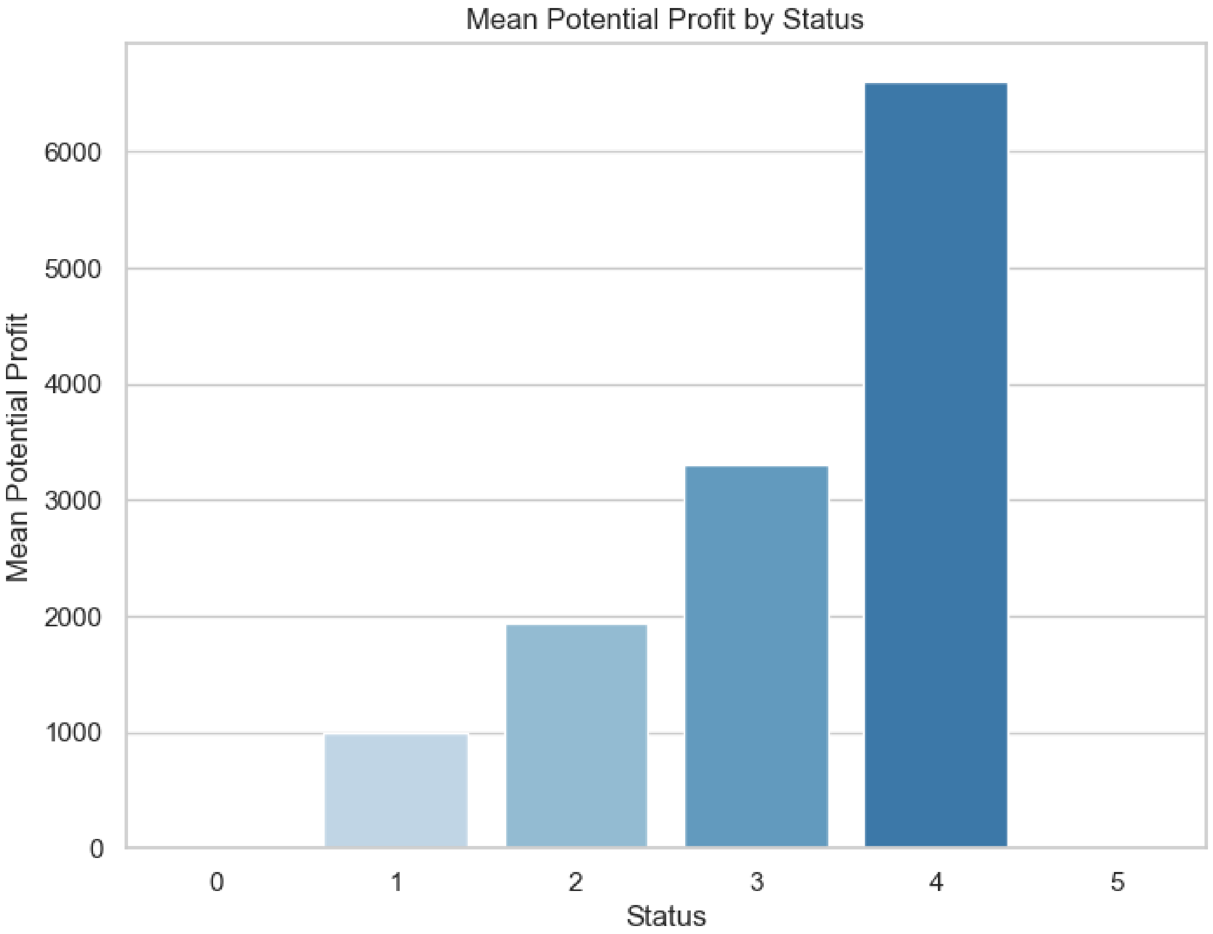
In [101…

```
# Calculate mean profit and mean potential loss for each Risk_Degree valu
mean_profit = df.groupby('Risk_Degree')['Profit'].mean()
mean_potential_loss = df.groupby('Risk_Degree')['Potential Loss'].mean()

# Create bar chart with two bars per Risk_Degree value
fig, ax = plt.subplots()
ax.bar(mean_profit.index, mean_profit.values, width=0.4, label='Mean Prof
ax.bar(mean_potential_loss.index, mean_potential_loss.values, width=-0.4,
ax.set_xlabel('Risk Degree')
ax.set_ylabel('Amount (£)')
ax.set_title('Mean Profit and Mean Potential Loss by Risk Degree')
ax.legend()
plt.show()
```



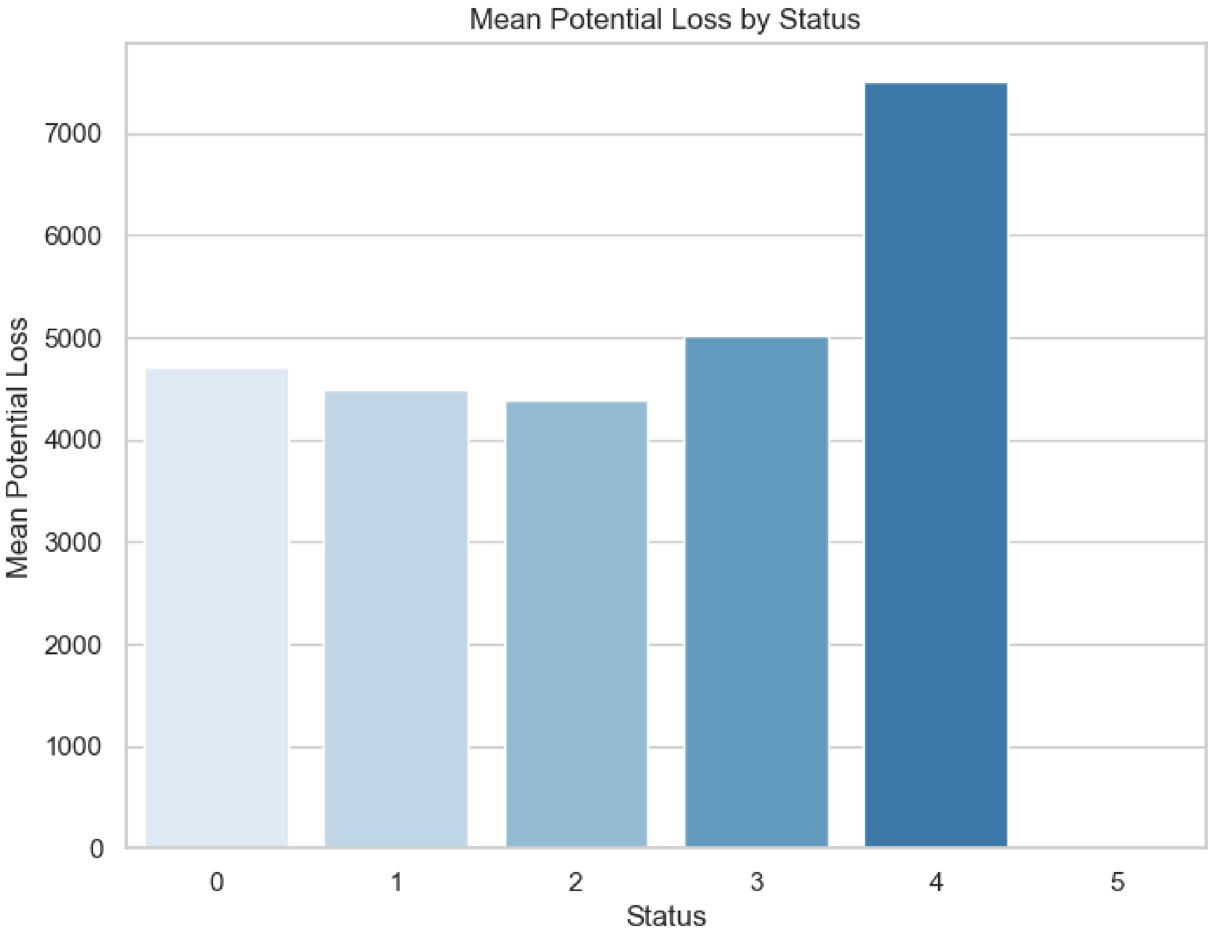
In [102…

```
potential_profit_by_status = df.groupby('STATUS')['Profit'].mean().reset_
import seaborn as sns
import matplotlib.pyplot as plt

sns.set_style('whitegrid')
plt.figure(figsize=(8,6))
sns.barplot(x='STATUS', y='Profit', data=potential_profit_by_status, pale
plt.title('Mean Potential Profit by Status')
plt.xlabel('Status')
plt.ylabel('Mean Potential Profit')
plt.show()
```



In [103…

```
potential_loss_by_status = df.groupby('STATUS')['Potential Loss'].mean().
import seaborn as sns
import matplotlib.pyplot as plt

sns.set_style('whitegrid')
plt.figure(figsize=(8,6))
sns.barplot(x='STATUS', y='Potential Loss', data=potential_loss_by_status
plt.title('Mean Potential Loss by Status')
plt.xlabel('Status')
plt.ylabel('Mean Potential Loss')
plt.show()
```



```
In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```