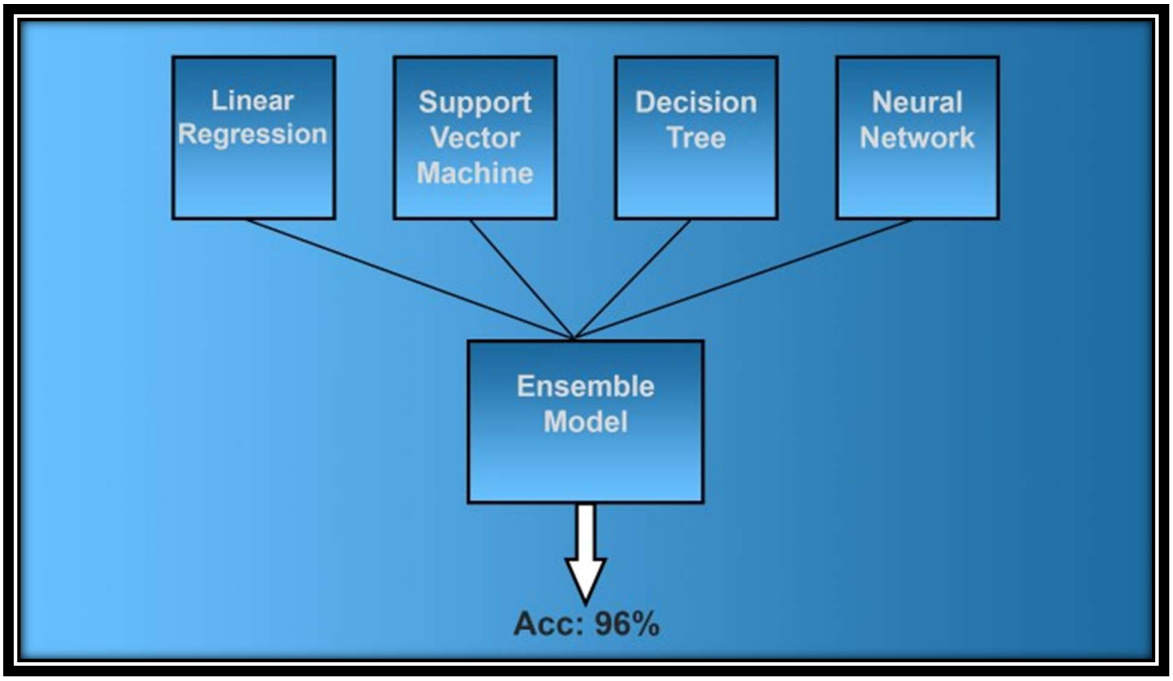
# ENSEMBLE MODEL




Predicting whether the client of a certain bank in Taiwan will default to the next month or not.

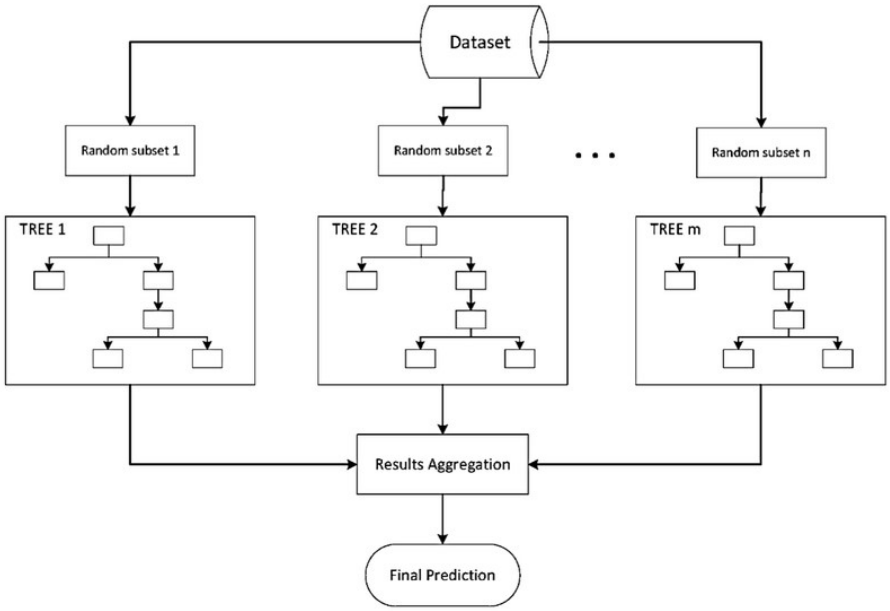
| | |
|---|---|
| Date | 14/04/2020 |
| Name | Vinit Ravichandran Iyer |
| | Ashmitha Nagesh |

# Content List

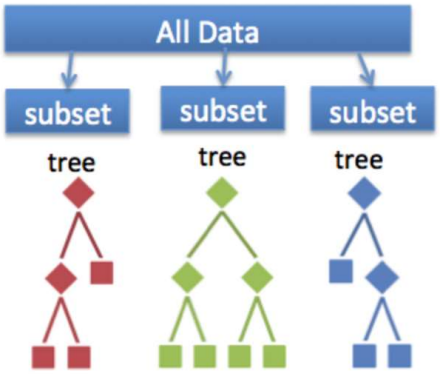
# Introduction

**Ensemble methods** is a machine learning technique that combines several base models in order to produce one optimal predictive model. There are different types of Ensemble models, namely:

- BAGGing, or Bootstrap AGGregating. **BAGG**ing gets its name because it combines Bootstrapping and Aggregation to form one ensemble model. Given a sample of data, multiple bootstrapped subsamples are pulled. A Decision Tree is formed on each of the bootstrapped subsamples. After each subsample Decision Tree has been formed, an algorithm is used to aggregate over the Decision Trees to form the most efficient predictor.




- Random Forest Models. Random Forest Models can be thought of as BAGGing, with a slight tweak. When deciding where to split and how to make decisions, BAGGed Decision Trees have the full disposal of features to choose from. Therefore, although the bootstrapped samples may be slightly different, the data is largely going to break off at the same features throughout each model. In contrary, Random Forest models decide where to split based on a random selection of features. Rather than splitting at similar features at each node throughout, Random Forest models implement a level of differentiation because each tree will split based on different features. This level of differentiation provides a greater ensemble to aggregate over, ergo producing a more accurate predictor.




A random forest takes a random subset of features from the data, and creates n random trees from each subset. Trees are aggregated together at end.

The goal of any machine learning problem is to find a single model that will best predict our wanted outcome. Rather than making one model and hoping this model is the best/most accurate predictor we can make, ensemble methods take a myriad of models into account, and average those models to produce one final model. It is important to note that Decision Trees are not the only form of ensemble methods, just the most popular and relevant in Data Science today.

In the model described hereafter, we have found the AUC. An evaluation metric that considers all possible classification thresholds. The Area Under the ROC curve is the probability that a classifier will be more confident that a randomly chosen positive example is actually positive than that a randomly chosen negative example is positive.

We investigated the data, checking for data unbalancing, visualizing the features and understanding the relationship between different features. We Concluded with **Random Forest Classifier**, for which we obtained an AUC score of **0.65**.

# Source Code

1. Importing Packages

| | |
|---|---|
| Input | import pandas as pd<br>import numpy as np<br>import seaborn as sea<br>import matplotlib.pyplot as plt<br><br>import gc<br>from datetime import datetime<br>from sklearn.model_selection import train_test_split<br>from sklearn.metrics import roc_auc_score<br>from sklearn.ensemble import RandomForestClassifier |

2. Importing the Data

| | |
|---|---|
| Input | data = pd.read_excel("default of credit card clients.xls", header = 1)<br><br>data.head() |

Output

| | ID | LIMIT_BAL | SEX | EDUCATION | MARRIAGE | AGE | PAY_0 | PAY_2 | PAY_3 | PAY_4 | ... | BILL_AMT4 | BILL_AMT5 | BILL_AMT6 | PAY_AMT1 | PAY_AMT2 | PAY_ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 20000 | 2 | 2 | 1 | 24 | 2 | 2 | -1 | -1 | ... | 0 | 0 | 0 | 0 | 689 | |
| 1 | 2 | 120000 | 2 | 2 | 2 | 26 | -1 | 2 | 0 | 0 | ... | 3272 | 3455 | 3261 | 0 | 1000 | |
| 2 | 3 | 90000 | 2 | 2 | 2 | 34 | 0 | 0 | 0 | 0 | ... | 14331 | 14948 | 15549 | 1518 | 1500 | |
| 3 | 4 | 50000 | 2 | 2 | 1 | 37 | 0 | 0 | 0 | 0 | ... | 28314 | 28959 | 29547 | 2000 | 2019 | |
| 4 | 5 | 50000 | 1 | 2 | 1 | 57 | -1 | 0 | -1 | 0 | ... | 20940 | 19146 | 19131 | 2000 | 36681 | |

5 rows × 25 columns

3. Knowing the Data

| | |
|---|---|
| Input | data.info() |
| Output | <class 'pandas.core.frame.DataFrame'><br>RangeIndex: 30000 entries, 0 to 29999<br>Data columns (total 25 columns):<br># Column Non-Null Count Dtype<br>--- ------ -------------- -----<br>0 ID 30000 non-null int64<br>1 LIMIT_BAL 30000 non-null int64<br>2 SEX 30000 non-null int64<br>3 EDUCATION 30000 non-null int64<br>4 MARRIAGE 30000 non-null int64<br>5 AGE 30000 non-null int64<br>6 PAY_0 30000 non-null int64<br>7 PAY_2 30000 non-null int64<br>8 PAY_3 30000 non-null int64<br>9 PAY_4 30000 non-null int64 |

| | |
|---|---|
| | 10 PAY_5 30000 non-null int64<br>11 PAY_6 30000 non-null int64<br>12 BILL_AMT1 30000 non-null int64<br>13 BILL_AMT2 30000 non-null int64<br>14 BILL_AMT3 30000 non-null int64<br>15 BILL_AMT4 30000 non-null int64<br>16 BILL_AMT5 30000 non-null int64<br>17 BILL_AMT6 30000 non-null int64<br>18 PAY_AMT1 30000 non-null int64<br>19 PAY_AMT2 30000 non-null int64<br>20 PAY_AMT3 30000 non-null int64<br>21 PAY_AMT4 30000 non-null int64<br>22 PAY_AMT5 30000 non-null int64<br>23 PAY_AMT6 30000 non-null int64<br>24 default payment next month 30000 non-null int64<br>dtypes: int64(25)<br>memory usage: 5.7 MB |
| Input | data.describe() |

Output

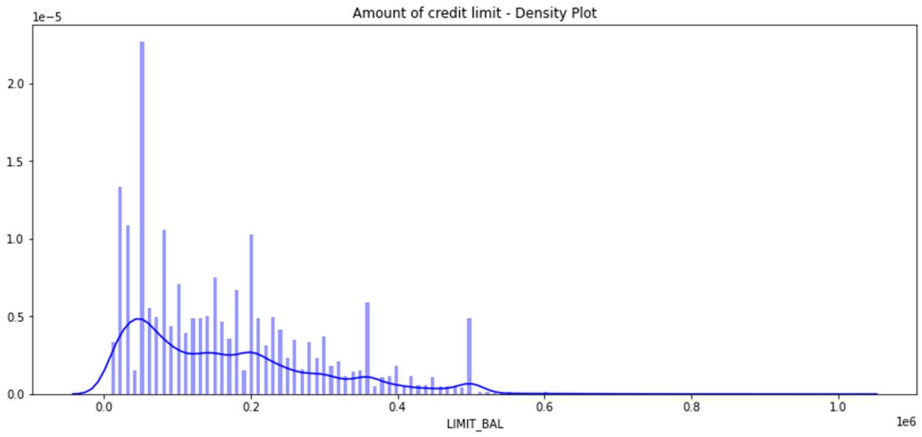
| | ID | LIMIT_BAL | SEX | EDUCATION | MARRIAGE | AGE | PAY_0 | PAY_2 | PAY_3 | PAY_4 | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| count | 30000.000000 | 30000.000000 | 30000.000000 | 30000.000000 | 30000.000000 | 30000.000000 | 30000.000000 | 30000.000000 | 30000.000000 | 30000.000000 | ... |
| mean | 15000.500000 | 167484.322667 | 1.603733 | 1.853133 | 1.551867 | 35.485500 | -0.016700 | -0.133767 | -0.166200 | -0.220667 | ... |
| std | 8660.398374 | 129747.661567 | 0.489129 | 0.790349 | 0.521970 | 9.217904 | 1.123802 | 1.197186 | 1.196868 | 1.169139 | ... |
| min | 1.000000 | 10000.000000 | 1.000000 | 0.000000 | 0.000000 | 21.000000 | -2.000000 | -2.000000 | -2.000000 | -2.000000 | ... |
| 25% | 7500.750000 | 50000.000000 | 1.000000 | 1.000000 | 1.000000 | 28.000000 | -1.000000 | -1.000000 | -1.000000 | -1.000000 | ... |
| 50% | 15000.500000 | 140000.000000 | 2.000000 | 2.000000 | 2.000000 | 34.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... |
| 75% | 22500.250000 | 240000.000000 | 2.000000 | 2.000000 | 2.000000 | 41.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... |
| max | 30000.000000 | 1000000.000000 | 2.000000 | 6.000000 | 3.000000 | 79.000000 | 8.000000 | 8.000000 | 8.000000 | 8.000000 | ... |

8 rows × 25 columns

- Checking the amount of Credit Limit

| | |
|---|---|
| Input | plt.figure(figsize = (14,6))<br>plt.title('Amount of credit limit - Density Plot')<br>sea.set_color_codes("pastel")<br>sea.distplot(data['LIMIT_BAL'],kde=True,bins=200, color="blue")<br>plt.show() |

Output




Largest group of amount of credit limit is apparently for amount of 50K.

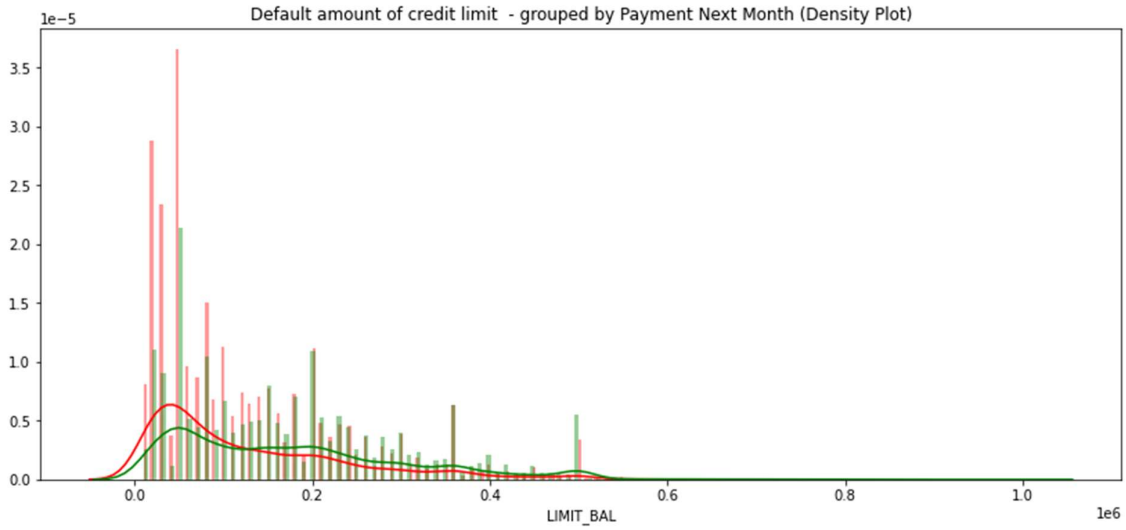
| Input | data['LIMIT_BAL'].value_counts().head(5) |
|---|---|
| Output | 50000 3365<br>20000 1976<br>30000 1610<br>80000 1567<br>200000 1528<br>Name: LIMIT_BAL, dtype: int64 |

This proves that the largest group of amount of credit limit is 50k

- Visualizing the amount of Credit Limit grouped by Default payment of the next month

| Input | class_0 = data.loc[data['default payment next month'] == 0]["LIMIT_BAL"]<br>class_1 = data.loc[data['default payment next month'] == 1]["LIMIT_BAL"]<br>plt.figure(figsize = (14,6))<br>plt.title('Default amount of credit limit - grouped by Payment Next Month (Density Plot)')<br>sea.set_color_codes("pastel")<br>sea.distplot(class_1,kde=True,bins=200, color="red")<br>sea.distplot(class_0,kde=True,bins=200, color="green")<br>plt.show() |
|---|---|

Output

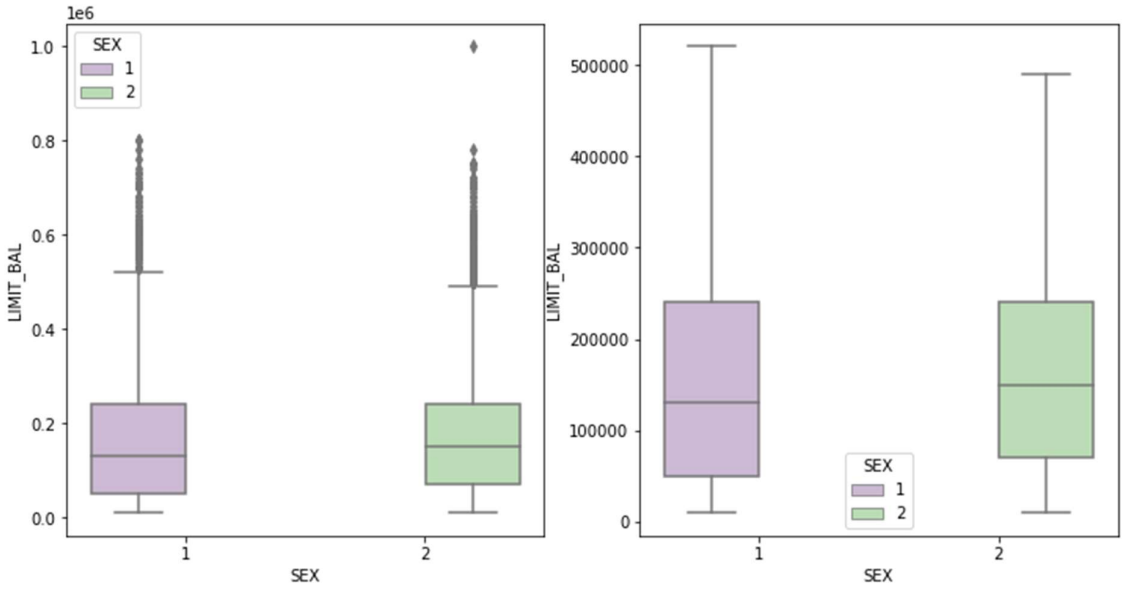



Most of defaults are for credit limits 0-100,000 (and density for this interval is larger for defaults than for non-defaults). Larger defaults number are for the amounts of 50,000, 20,000 and 30,000.

- Visualizing the Credit Limit relationship with respect to Gender

| Input | fig, (axmale, axfemale) = plt.subplots(ncols=2, figsize=(12,6))<br>s = sea.boxplot(ax = axmale, x="SEX", y="LIMIT_BAL",<br>hue="SEX",data=data, palette="PRGn",showfliers=True)<br>s = sea.boxplot(ax = axfemale, x="SEX", y="LIMIT_BAL",<br>hue="SEX",data=data, palette="PRGn",showfliers=False)<br>plt.show(); |
|---|---|

Output




- 1 = Male 2 = Female
- The limit credit amount is quite balanced between sexes.
- The males have a slightly smaller Q2 and larger Q3 and Q4 and a lower mean.
- The female have a larger outlier max value.

4. Preparing the Data
    - Checking for missing data

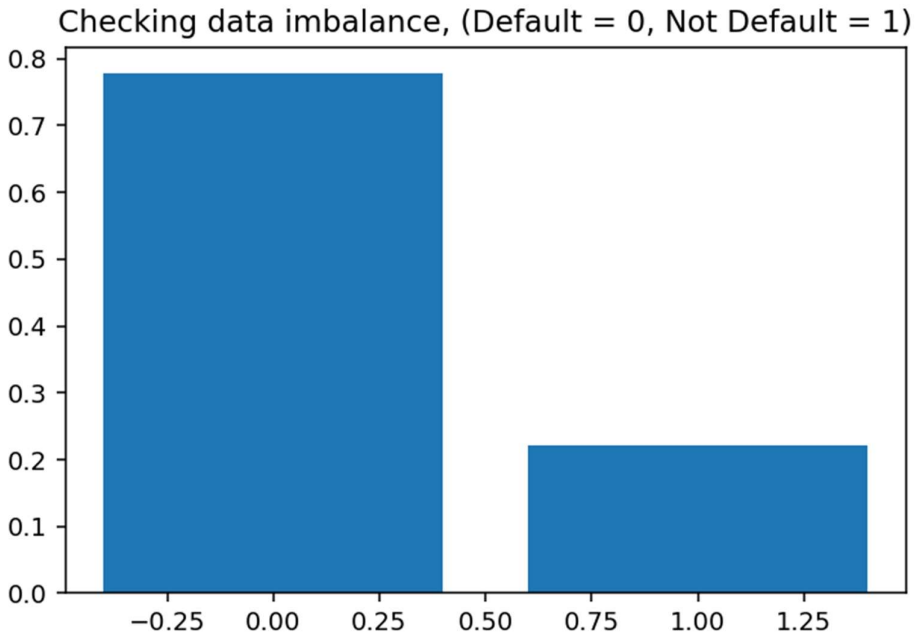
| Input | total = data.isnull().sum().sort_values(ascending = False)<br>print(total) |
|---|---|
| Output | default payment next month 0<br>PAY_6 0<br>LIMIT_BAL 0<br>SEX 0<br>EDUCATION 0<br>MARRIAGE 0<br>AGE 0<br>PAY_0 0<br>PAY_2 0<br>PAY_3 0<br>PAY_4 0<br>PAY_5 0<br>BILL_AMT1 0<br>PAY_AMT6 0<br>BILL_AMT2 0<br>BILL_AMT3 0<br>BILL_AMT4 0<br>BILL_AMT5 0<br>BILL_AMT6 0<br>PAY_AMT1 0<br>PAY_AMT2 0<br>PAY_AMT3 0<br>PAY_AMT4 0<br>PAY_AMT5 0 |

| | |
|---|---|
| | ID 0<br>dtype: int64 |

- Checking for data imbalance

| | |
|---|---|
| Input | temp = data["default payment next month"].value_counts(normalize=True)<br>df = pd.DataFrame({'default payment next month': temp.index,'values': temp.values})<br><br>plt.figure(dpi = 140)<br>plt.title('Checking data imbalance, (Default = 0, Not Default = 1)')<br>plt.bar(df['default payment next month'], df['values'])<br>plt.show() |

Output

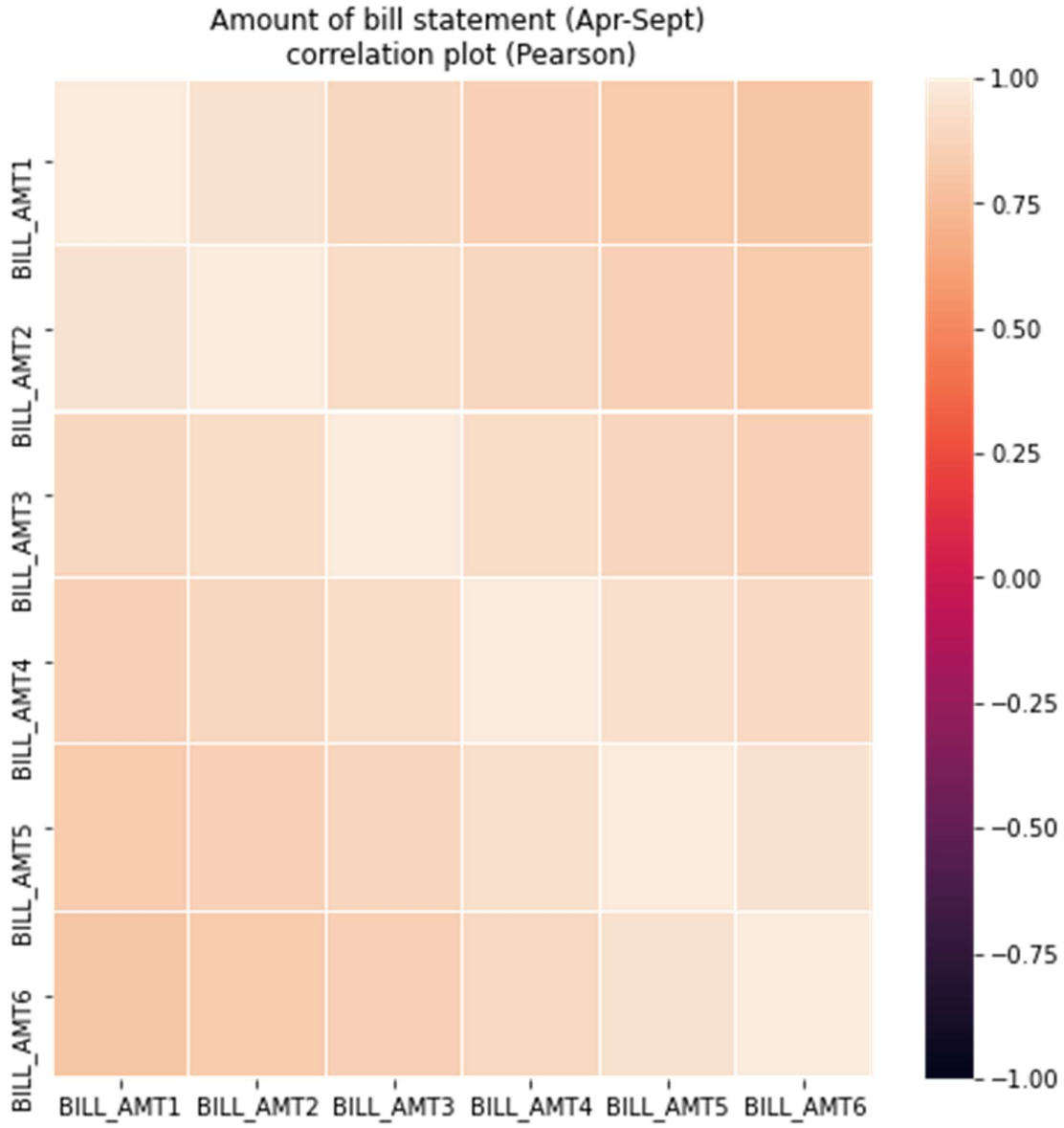



A minimum of 22% of the total clients will defualt into next month. The data does not have a large imbalance with respect to the target value - Default payment for next month.

5. Feature Correlation Matrix

| | |
|---|---|
| Input | var = ['BILL_AMT1','BILL_AMT2','BILL_AMT3','BILL_AMT4','BILL_AMT5', 'BILL_AMT6']<br><br>plt.figure(figsize = (8,8))<br>plt.title('Amount of bill statement (Apr-Sept) \ncorrelation plot (Pearson)')<br>corr = data[var].corr()<br>sea.heatmap(corr,xticklabels=corr.columns,yticklabels=corr.columns,linewidths=.1,vmin=-1, vmax=1)<br>plt.show() |

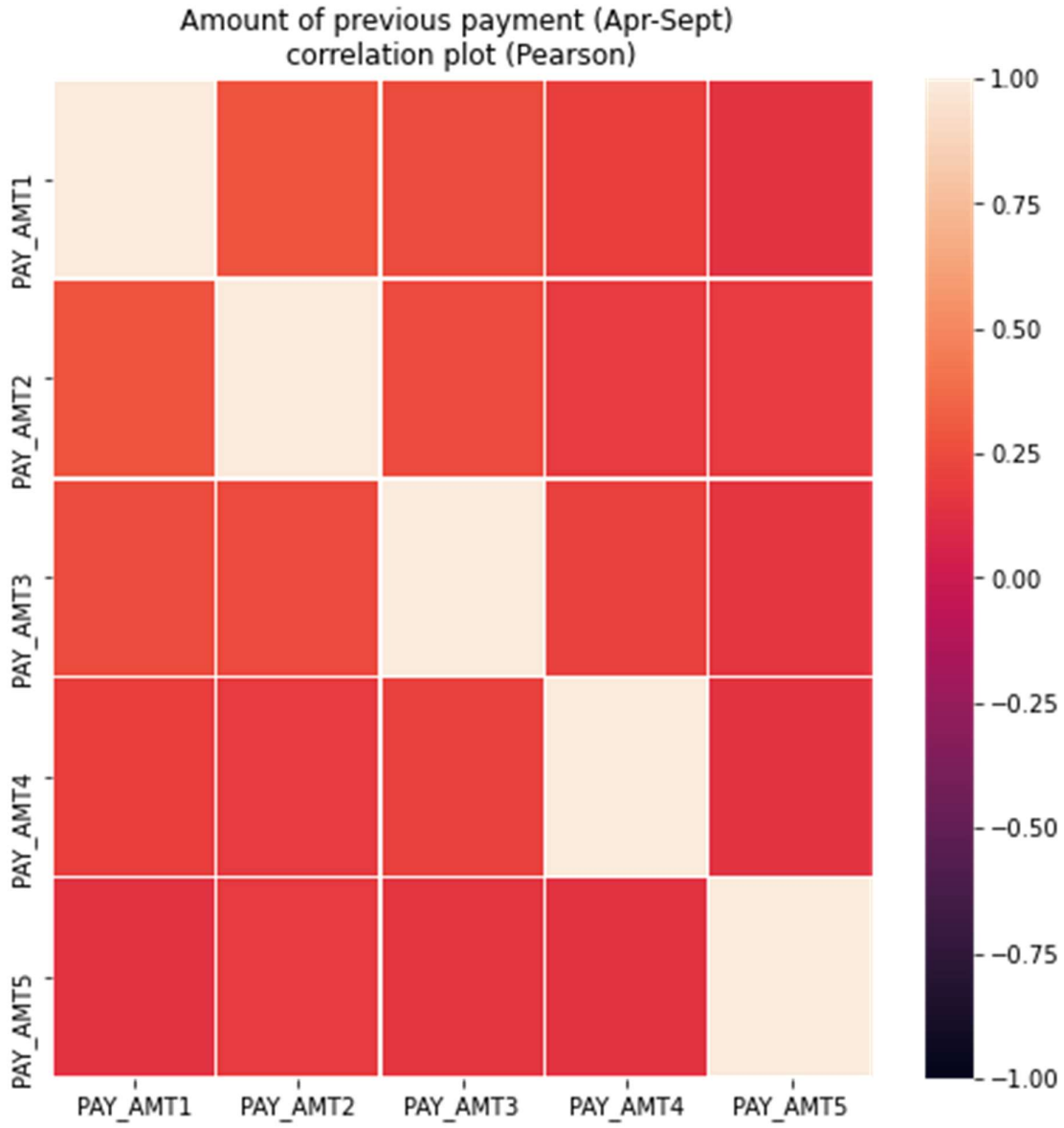
Output




Correlation is decreasing with distance between months. Lowest correlations are between Sept-April.

| | |
|---|---|
| Input | var = ['PAY_AMT1', 'PAY_AMT2', 'PAY_AMT3', 'PAY_AMT4', 'PAY_AMT5']<br><br>plt.figure(figsize = (8,8))<br>plt.title('Amount of previous payment (Apr-Sept) \ncorrelation plot (Pearson)')<br>corr = data[var].corr()<br>sea.heatmap(corr,xticklabels=corr.columns,yticklabels=corr.columns,linewidths=.1,vmin=-1, vmax=1)<br>plt.show() |

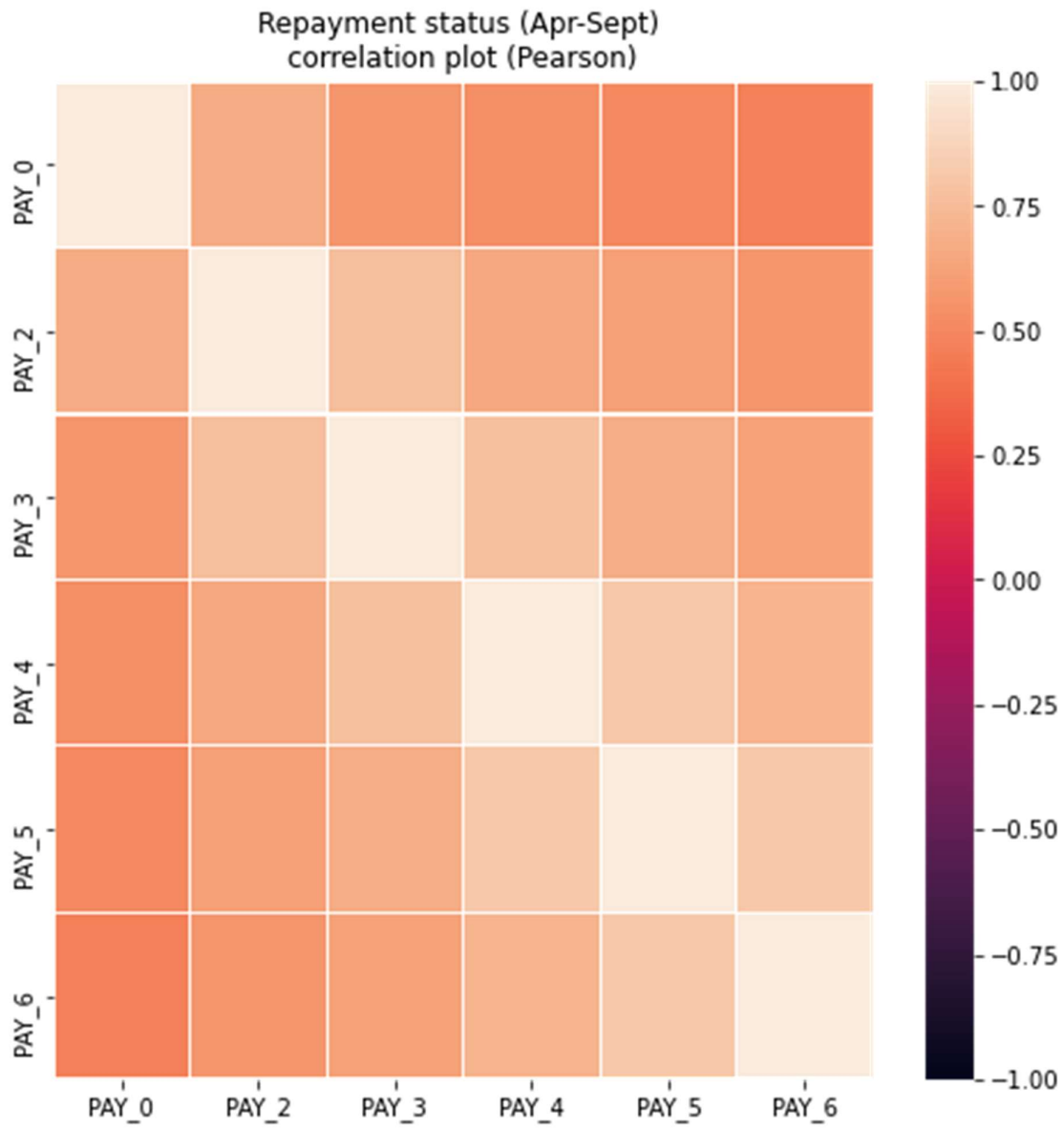
Output




There are no correlations between amounts of previous payments for April-Sept 2005.

| | |
|---|---|
| Input | var = ['PAY_0','PAY_2', 'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6']<br><br>plt.figure(figsize = (8,8))<br>plt.title('Repayment status (Apr-Sept) \ncorrelation plot (Pearson)')<br>corr = data[var].corr()<br>sea.heatmap(corr,xticklabels=corr.columns,yticklabels=corr.columns,linewidths=.1,vmin=-1, vmax=1)<br>plt.show() |

Output




Correlation is decreasing with distance between months. Lowest correlations are between Sept-April.

6. Visualizing Sex, Education, Age and Marriage variables

Marriage status meaning is:

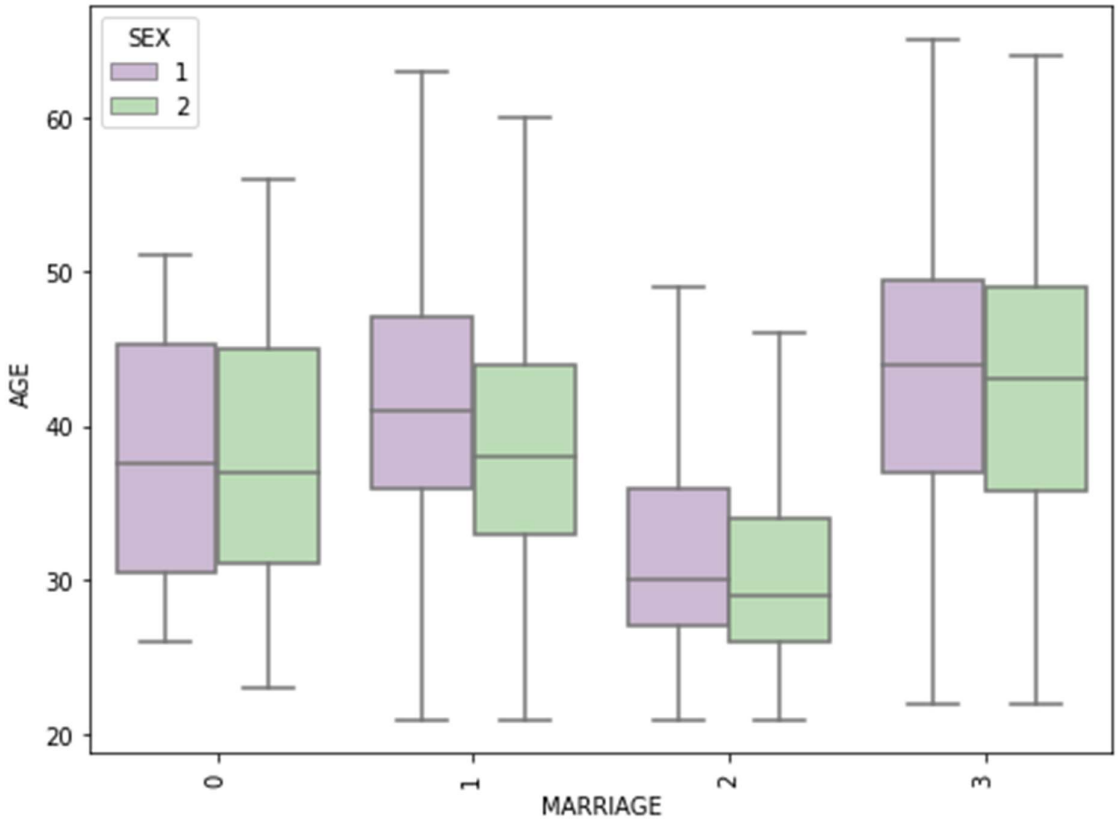
- 0 : unknown (let's consider as others as well)
- 1 : married
- 2 : single
- 3 : others

Sex meaning is:

- 1 : male
- 2 : female

| | |
|---|---|
| Input | def boxplot_variation(feature1, feature2, feature3, width=16):<br>fig, ax1 = plt.subplots(ncols=1, figsize=(width,6))<br>s = sea.boxplot(ax = ax1, x=feature1, y=feature2, hue=feature3,<br>data=data, palette="PRGn",showfliers=False)<br>s.set_xticklabels(s.get_xticklabels(),rotation=90)<br>plt.show(); |
| Input | boxplot_variation('MARRIAGE','AGE', 'SEX',8) |

Output



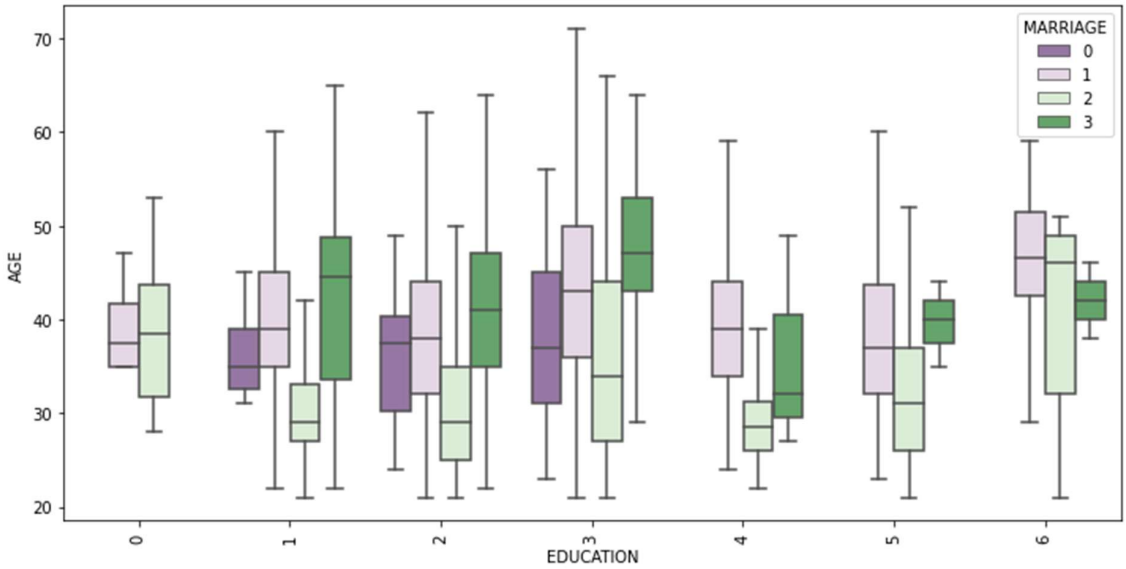

It looks like Married status 3 (others), with mean values over 40 and Q4 values over 60 means mostly vidowed or divorced whilst Married status 0 could be not specified or divorced, as Q1 values are above values for married of both sexes.

Married males have mean age above married women. Unmarried males have mean value for age above unmarried women as well but closer. Q3 abd Q4 values for married man are above corresponding values for married women.
Education status meaning is:

- 1 : graduate school
- 2 : university
- 3 : high school
- 4 : others
- 5 : unknown
- 6 : unknow

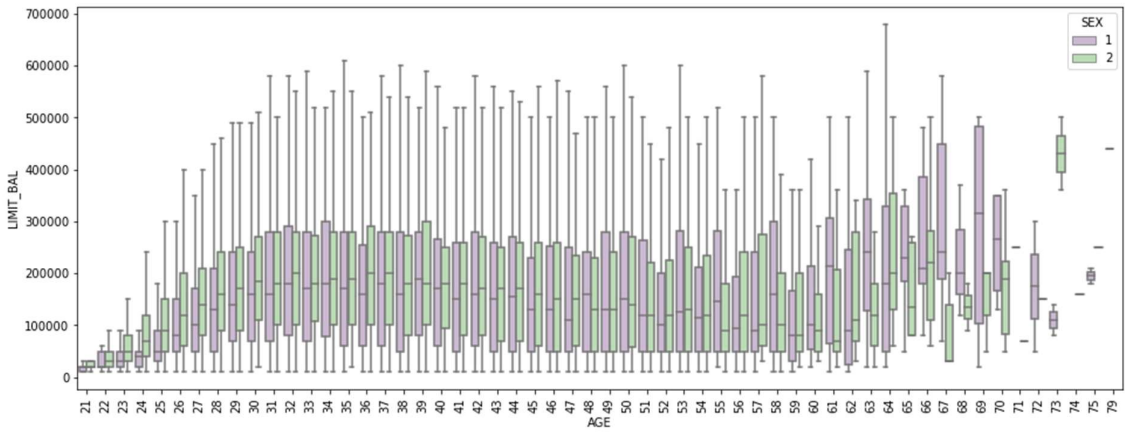
| | |
|---|---|
| Input | boxplot_variation('EDUCATION','AGE', 'MARRIAGE',12) |

Output




| Input | boxplot_variation('AGE','LIMIT_BAL', 'SEX',16) |
|---|---|

Output

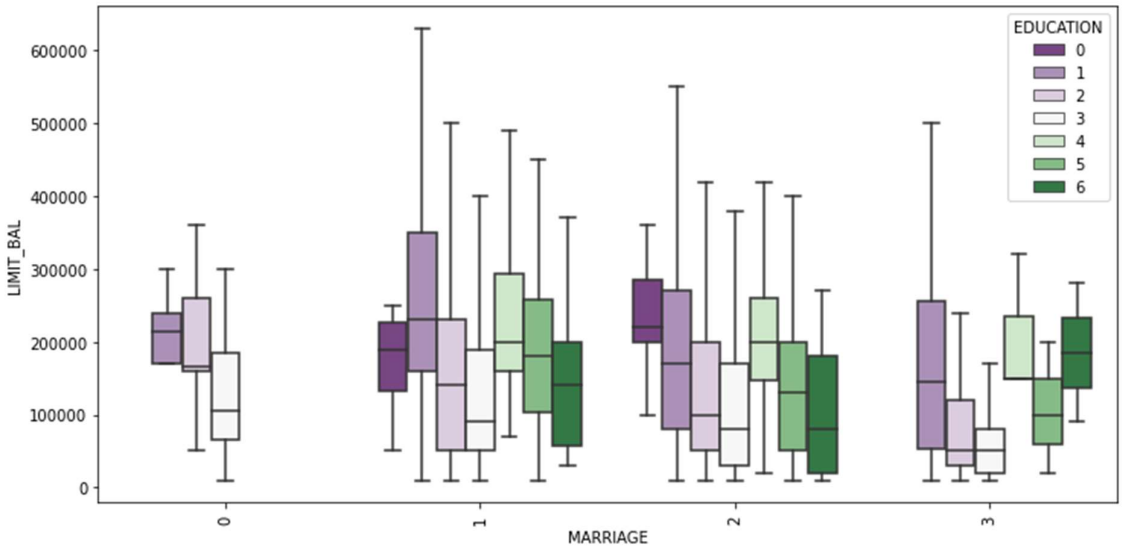



Above plot is for Credit Limit amount distribution grouped by Age and Sex
Mean, Q3 and Q4 values are increasing for both male and female with age until aroung 35 years and then they are oscilating and get to a maximum of Q4 for males at age 64.

Mean values are generally smaller for males than for females, with few exceptions, for example at age 39, 48, until approximately 60, where mean values for males are generally larger than for females.

| Input | boxplot_variation('MARRIAGE','LIMIT_BAL', 'EDUCATION',12) |
|---|---|

Output




Above plot is for Credit Limit amount distribution grouped by Marriage status and Education level

7. Splitting the data for Training and Test data

| Input | target = 'default.payment.next.month'<br>predictors = [ 'LIMIT_BAL', 'SEX', 'EDUCATION', 'MARRIAGE', 'AGE',<br>'PAY_0', 'PAY_2', 'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6',<br>'BILL_AMT1','BILL_AMT2', 'BILL_AMT3', 'BILL_AMT4',<br>'BILL_AMT5', 'BILL_AMT6',<br>'PAY_AMT1', 'PAY_AMT2', 'PAY_AMT3', 'PAY_AMT4',<br>'PAY_AMT5', 'PAY_AMT6']<br><br>train_df, val_df = train_test_split(data, test_size=0.25, random_state=42,<br>shuffle=True ) |
|---|---|

8. Random Forest Classifier

We will use as validation criterion GINI, which formula is GINI = 2 * (AUC) - 1, where AUC is the Receiver Operating Characteristic - Area Under Curve (ROC-AUC) . Number of estimators is set to 100 and number of parallel jobs is set to 4.

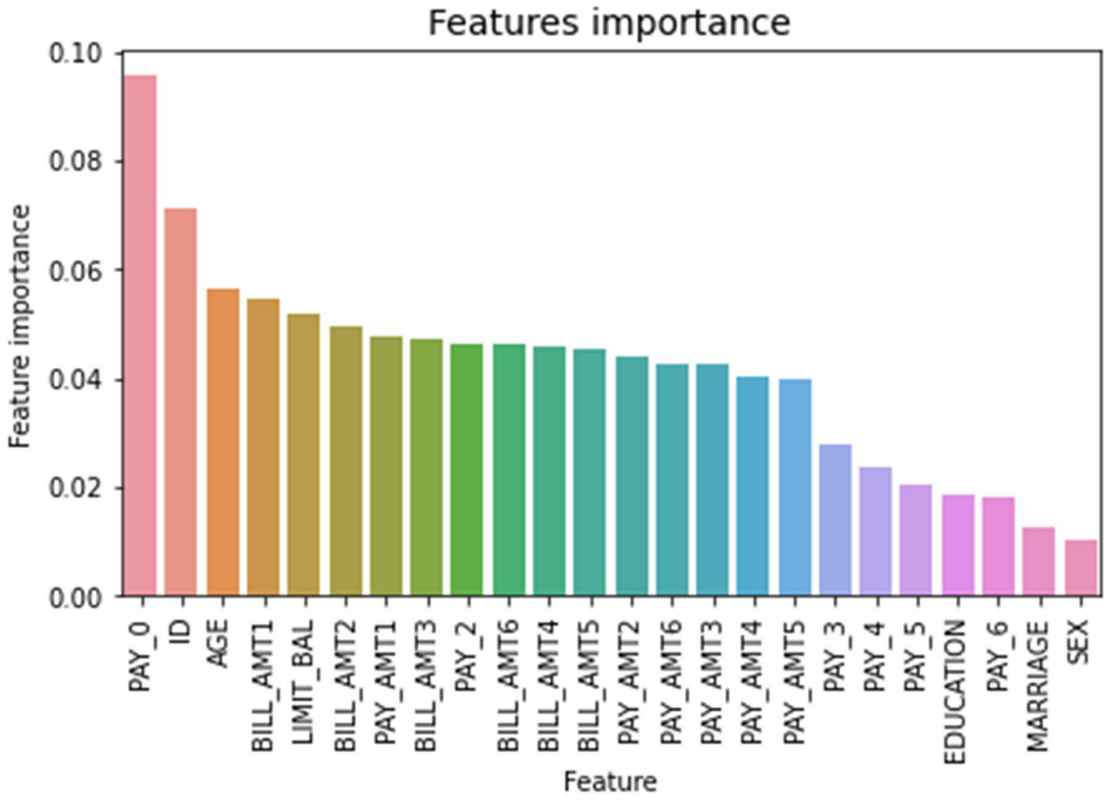
| Input | clf = RandomForestClassifier(n_jobs=-1,<br>random_state=42,<br>criterion='gini',<br>n_estimators=350,<br>verbose=False)<br><br>predictors = train_df.drop(columns=['default payment next<br>month']).columns.values<br>target = 'default payment next month' |
|---|---|

| | |
|---|---|
| Input | clf.fit(train_df[predictors].values, train_df[target].values) |
| Output | `RandomForestClassifier(n_estimators=350, n_jobs=-1, random_state=42,`<br>`verbose=False)` |
| Input | preds = clf.predict(val_df[predictors]) |

- Features Importance

| | |
|---|---|
| Input | tmp = pd.DataFrame({'Feature': predictors, 'Feature importance': clf.feature_importances_})<br>tmp = tmp.sort_values(by='Feature importance',ascending=False)<br>plt.figure(figsize = (7,4))<br>plt.title('Features importance',fontsize=14)<br>s = sea.barplot(x='Feature',y='Feature importance',data=tmp)<br>s.set_xticklabels(s.get_xticklabels(),rotation=90)<br>plt.show() |

Output

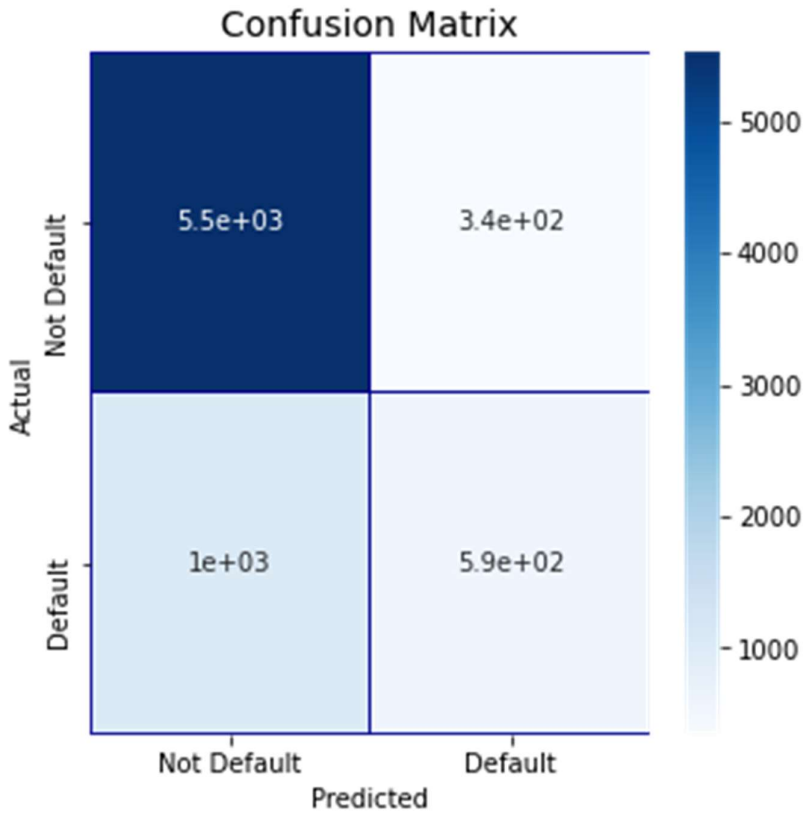



The most important features are PAY_0, AGE, BILL_AMT1, LIMIT_BAL, BILL_AMT2, BILL_AMT3.

- Confusion Matrix

| | |
|---|---|
| Input | cm = pd.crosstab(val_df[target].values, preds, rownames=['Actual'], colnames=['Predicted'])<br>fig, (ax1) = plt.subplots(ncols=1, figsize=(5,5))<br>sea.heatmap(cm,<br>xticklabels=['Not Default', 'Default'],<br>yticklabels=['Not Default', 'Default'],<br>annot=True,ax=ax1,<br>linewidths=.2,linecolor="Darkblue", cmap="Blues")<br>plt.title('Confusion Matrix', fontsize=14)<br>plt.show() |

Output



9. Random Forest with One Hot Encoder

| | |
|---|---|
| Input | cat_features = ['EDUCATION', 'SEX', 'MARRIAGE', 'PAY_0', 'PAY_2', 'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6']<br><br>train_df_bkp = train_df.copy()<br>val_df_bkp = val_df.copy()<br><br>train_f_df = pd.get_dummies(train_df_bkp, columns = cat_features)<br>val_f_df = pd.get_dummies(val_df_bkp, columns = cat_features)<br><br>print("Default of Credit Card Clients train data - rows:",train_f_df.shape[0]," columns:", train_f_df.shape[1])<br>print("Default of Credit Card Clients val data - rows:",val_f_df.shape[0]," columns:", val_f_df.shape[1]) |
| Output | `Default of Credit Card Clients train data - rows: 22500 columns: 93`<br>`Default of Credit Card Clients val data - rows: 7500 columns: 89` |
| Input | train_fa_df, val_fa_df = train_f_df.align(val_f_df, join='outer', axis=1, fill_value=0)<br><br>print("Default of Credit Card Clients train data - rows:",train_fa_df.shape[0]," columns:", train_fa_df.shape[1])<br>print("Default of Credit Card Clients val data - rows:",val_fa_df.shape[0]," columns:", val_fa_df.shape[1]) |
| Output | `Default of Credit Card Clients train data - rows: 22500 columns: 93`<br>`Default of Credit Card Clients val data - rows: 7500 columns: 93` |

| | |
|---|---|
| Input | train_fa_df.head(5) |

Output

| | AGE | BILL_AMT1 | BILL_AMT2 | BILL_AMT3 | BILL_AMT4 | BILL_AMT5 | BILL_AMT6 | EDUCATION_0 | EDUCATION_1 | EDUCATION_2 | ... | PAY_6_8 | PAY_AMT |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **21177** | 31 | 80928 | 82690 | 84462 | 86263 | 87238 | 89176 | 0 | 0 | 0 | ... | 0 | 400 |
| **23942** | 24 | 15730 | 16776 | 35036 | 14694 | 16914 | 14074 | 0 | 0 | 1 | ... | 0 | 131 |
| **1247** | 35 | 2667 | 2667 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | ... | 0 | 266 |
| **23622** | 40 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | ... | 0 | |
| **28454** | 36 | 68028 | 67864 | 59165 | 29314 | 28844 | 29443 | 0 | 0 | 1 | ... | 0 | 334 |

5 rows × 93 columns

| Input | train_fa_df.head(5) |
|---|---|

Output

| | AGE | BILL_AMT1 | BILL_AMT2 | BILL_AMT3 | BILL_AMT4 | BILL_AMT5 | BILL_AMT6 | EDUCATION_0 | EDUCATION_1 | EDUCATION_2 | ... | PAY_6_8 | PAY_AMT |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **2308** | 25 | 8864 | 10062 | 11581 | 12580 | 13716 | 14828 | 0 | 0 | 1 | ... | 0 | 150 |
| **22404** | 26 | 136736 | 125651 | 116684 | 101581 | 77741 | 77264 | 0 | 1 | 0 | ... | 0 | 448 |
| **23397** | 32 | 70122 | 69080 | 68530 | 69753 | 70111 | 70212 | 0 | 0 | 0 | ... | 0 | 243 |
| **25058** | 49 | 20678 | 18956 | 16172 | 16898 | 11236 | 6944 | 0 | 0 | 0 | ... | 0 | 161 |
| **2664** | 36 | 94228 | 47635 | 42361 | 19574 | 20295 | 19439 | 0 | 0 | 1 | ... | 0 | 200 |

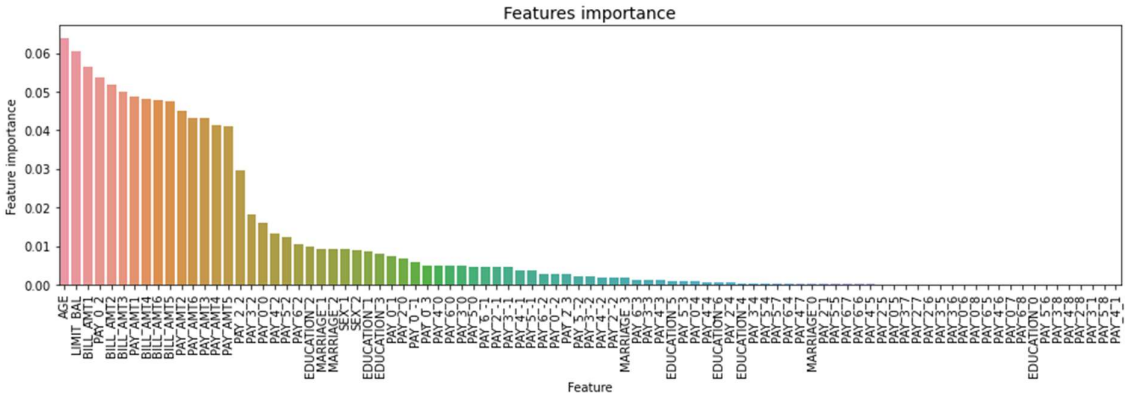
5 rows × 93 columns

| Input | target_f = 'default payment next month'<br>predictors_f = ['AGE', 'BILL_AMT1', 'BILL_AMT2', 'BILL_AMT3', 'BILL_AMT4',<br>'BILL_AMT5', 'BILL_AMT6', 'EDUCATION_0', 'EDUCATION_1',<br>'EDUCATION_2', 'EDUCATION_3', 'EDUCATION_4', 'EDUCATION_5',<br>'EDUCATION_6', 'LIMIT_BAL', 'MARRIAGE_0', 'MARRIAGE_1',<br>'MARRIAGE_2', 'MARRIAGE_3', 'PAY_0_-1', 'PAY_0_-2', 'PAY_0_0',<br>'PAY_0_1', 'PAY_0_2', 'PAY_0_3', 'PAY_0_4', 'PAY_0_5', 'PAY_0_6',<br>'PAY_0_7', 'PAY_0_8', 'PAY_2_-1', 'PAY_2_-2', 'PAY_2_0', 'PAY_2_1',<br>'PAY_2_2', 'PAY_2_3', 'PAY_2_4', 'PAY_2_5', 'PAY_2_6', 'PAY_2_7',<br>'PAY_2_8', 'PAY_3_-1', 'PAY_3_-2', 'PAY_3_0', 'PAY_3_1', 'PAY_3_2',<br>'PAY_3_3', 'PAY_3_4', 'PAY_3_5', 'PAY_3_6', 'PAY_3_7', 'PAY_3_8',<br>'PAY_4_-1', 'PAY_4_-2', 'PAY_4_0', 'PAY_4_1', 'PAY_4_2', 'PAY_4_3',<br>'PAY_4_4', 'PAY_4_5', 'PAY_4_6', 'PAY_4_7', 'PAY_4_8', 'PAY_5_-1',<br>'PAY_5_-2', 'PAY_5_0', 'PAY_5_2', 'PAY_5_3', 'PAY_5_4', 'PAY_5_5',<br>'PAY_5_6', 'PAY_5_7', 'PAY_5_8', 'PAY_6_-1', 'PAY_6_-2', 'PAY_6_0',<br>'PAY_6_2', 'PAY_6_3', 'PAY_6_4', 'PAY_6_5', 'PAY_6_6', 'PAY_6_7',<br>'PAY_6_8', 'PAY_AMT1', 'PAY_AMT2', 'PAY_AMT3', 'PAY_AMT4',<br>'PAY_AMT5', 'PAY_AMT6', 'SEX_1', 'SEX_2'] |
|---|---|
| Input | clf.fit(train_fa_df[predictors_f], train_df[target_f].values) |
| Output | `RandomForestClassifier(n_estimators=350, n_jobs=-1, random_state=42,`<br>`verbose=False)` |
| Input | preds = clf.predict(val_fa_df[predictors_f]) |

- Features Importance

| Input | tmp = pd.DataFrame({'Feature': predictors_f, 'Feature importance':<br>clf.feature_importances_})<br>tmp = tmp.sort_values(by='Feature importance',ascending=False)<br>plt.figure(figsize = (16,4))<br>plt.title('Features importance',fontsize=14)<br>s = sea.barplot(x='Feature',y='Feature importance',data=tmp)<br>s.set_xticklabels(s.get_xticklabels(),rotation=90)<br>plt.show() |
|---|---|

Output

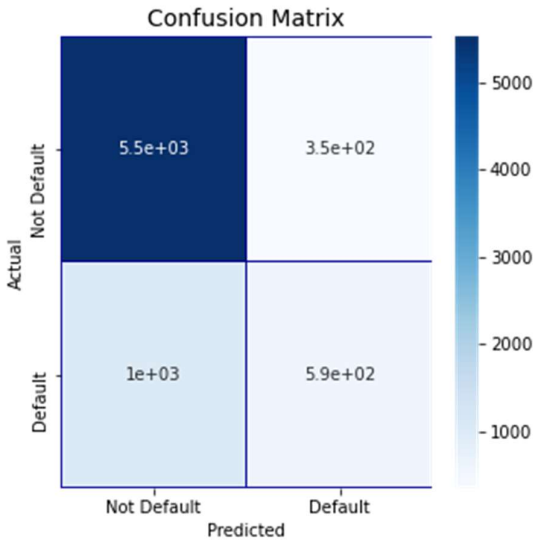



The most important features are AGE, LIMIT_BAL, BILL_AMT1, PAY_0_2, BILL_AMT2, BILL_AMT3.

- Confusion Matrix

| | |
|---|---|
| Input | cm = pd.crosstab(val_fa_df[target].values, preds, rownames=['Actual'], colnames=['Predicted'])<br>fig, (ax1) = plt.subplots(ncols=1, figsize=(5,5))<br>sea.heatmap(cm,<br>xticklabels=['Not Default', 'Default'],<br>yticklabels=['Not Default', 'Default'],<br>annot=True,ax=ax1,<br>linewidths=.2,linecolor="Darkblue", cmap="Blues")<br>plt.title('Confusion Matrix', fontsize=14)<br>plt.show() |

Output



# Output

| Input | roc_auc_score(val_df[target].values, preds) |
|---|---|
| Output | 0.6532579949015062 |

The ROC-AUC score obtained with Random Forest Classifier is 0.65.

We will use for Random Forest Classifier dummified variables for the categorical features.

| Input | _auc_score(val_fa_df[target].values, preds) |
|---|---|
| Output | 0.6507662549156908 |

With the dummified features, there is no improvement of the AUC score but the change is quite small. Still giving an AUC score of 0.65.

# Inference

We investigated the data, checking for data unbalancing, visualizing the features and understanding the relationship between different features. We Concluded with **Random Forest Classifier**, for which we obtained an AUC score of **0.65**.