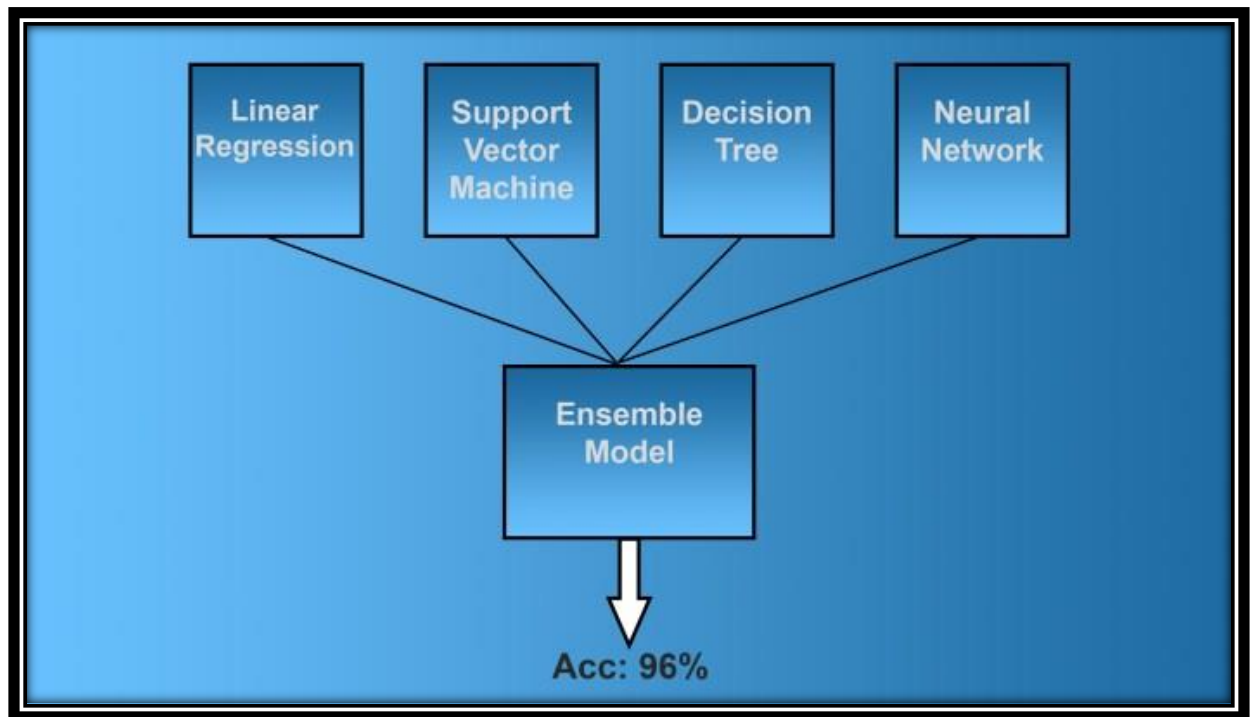


ENSEMBLE MODEL



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

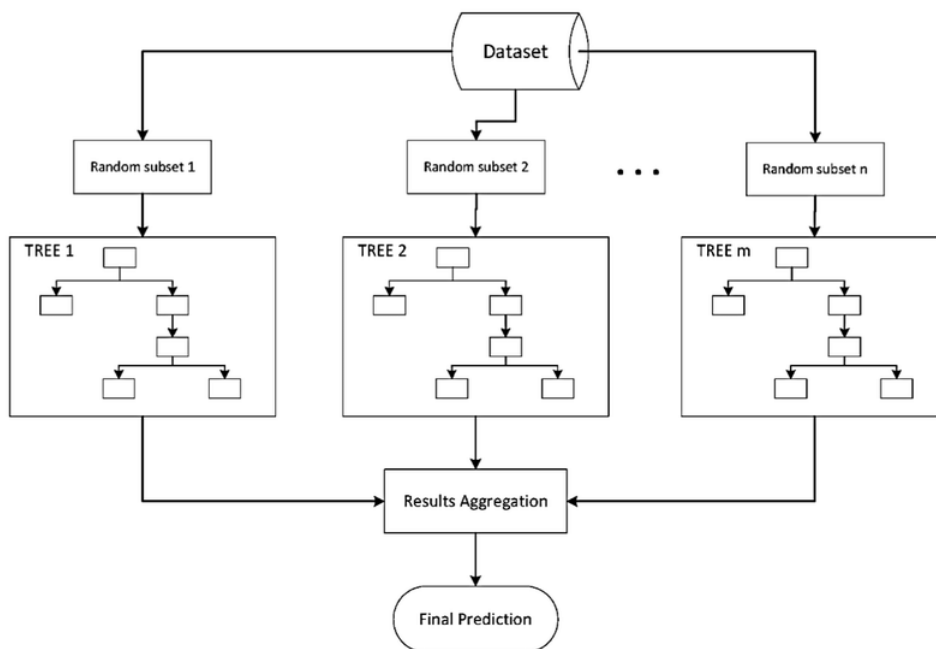
Content List

Sr No	Description	Page No
1	Introduction	3
2	Source Code	5
3	Coefficient Output	21
4	Inference	21

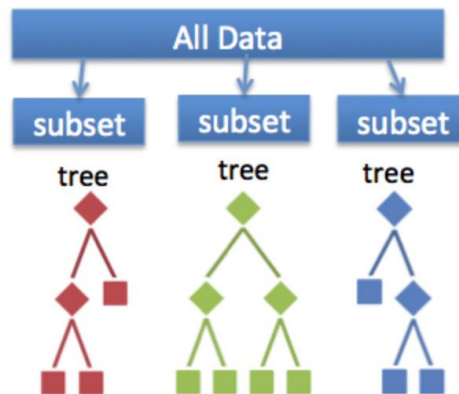
Introduction

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. **BAGGing** 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	<pre>import pandas as pd import numpy as np import seaborn as sea import matplotlib.pyplot as plt import gc from datetime import datetime from sklearn.model_selection import train_test_split from sklearn.metrics import roc_auc_score from sklearn.ensemble import RandomForestClassifier</pre>
-------	---

2. Importing the Data

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

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

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

Output

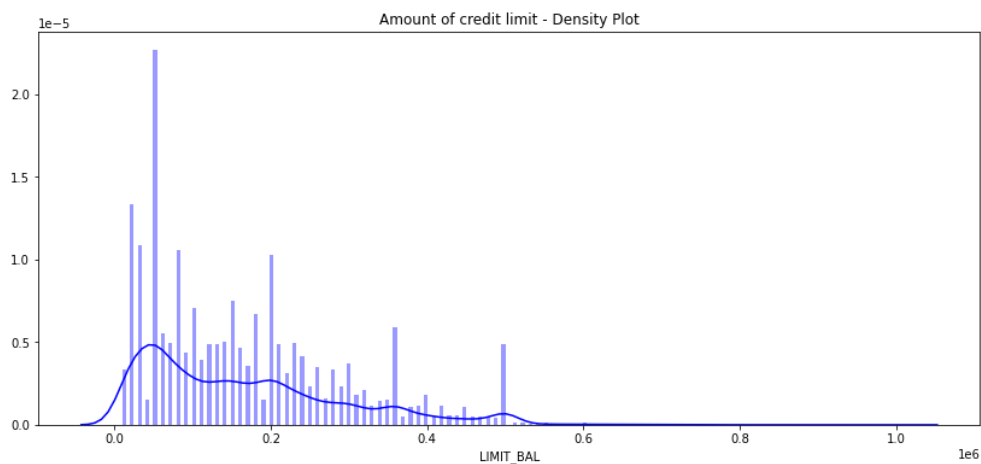
	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	...
count	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	...
mean	15000.500000	167484.322667	1.603733	1.853133	1.551867	35.485500	-0.016700	-0.133767	-0.166200	-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)) plt.title('Amount of credit limit - Density Plot') sea.set_color_codes("pastel") sea.distplot(data['LIMIT_BAL'],kde=True,bins=200, color="blue") plt.show()
-------	--

Output



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

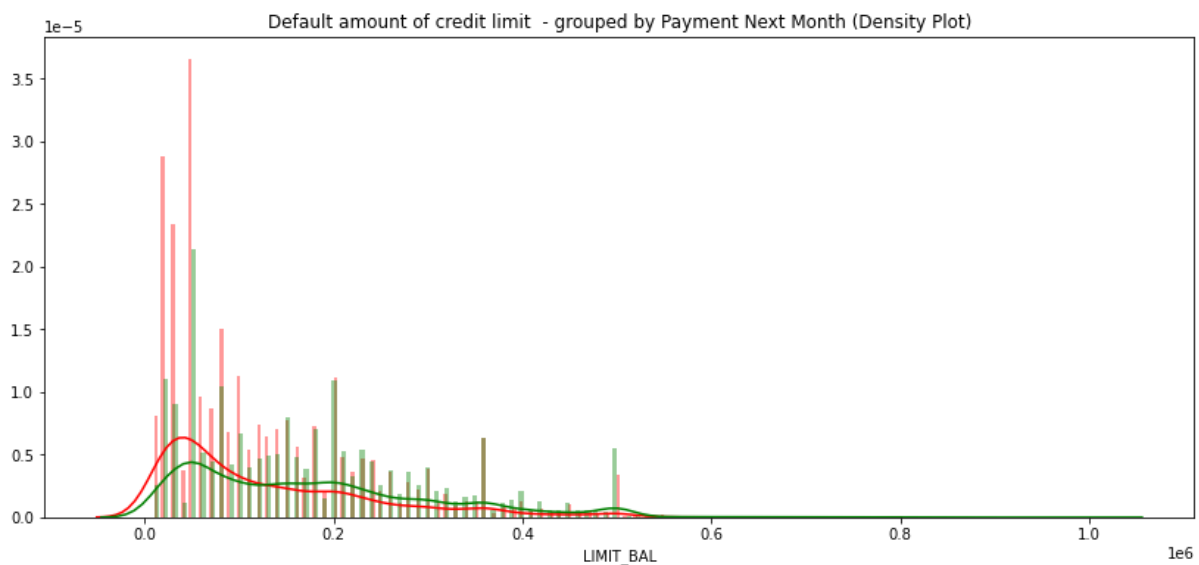
Input	<code>data['LIMIT_BAL'].value_counts().head(5)</code>
Output	<pre> 50000 3365 20000 1976 30000 1610 80000 1567 200000 1528 Name: LIMIT_BAL, dtype: int64 </pre>

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

Output

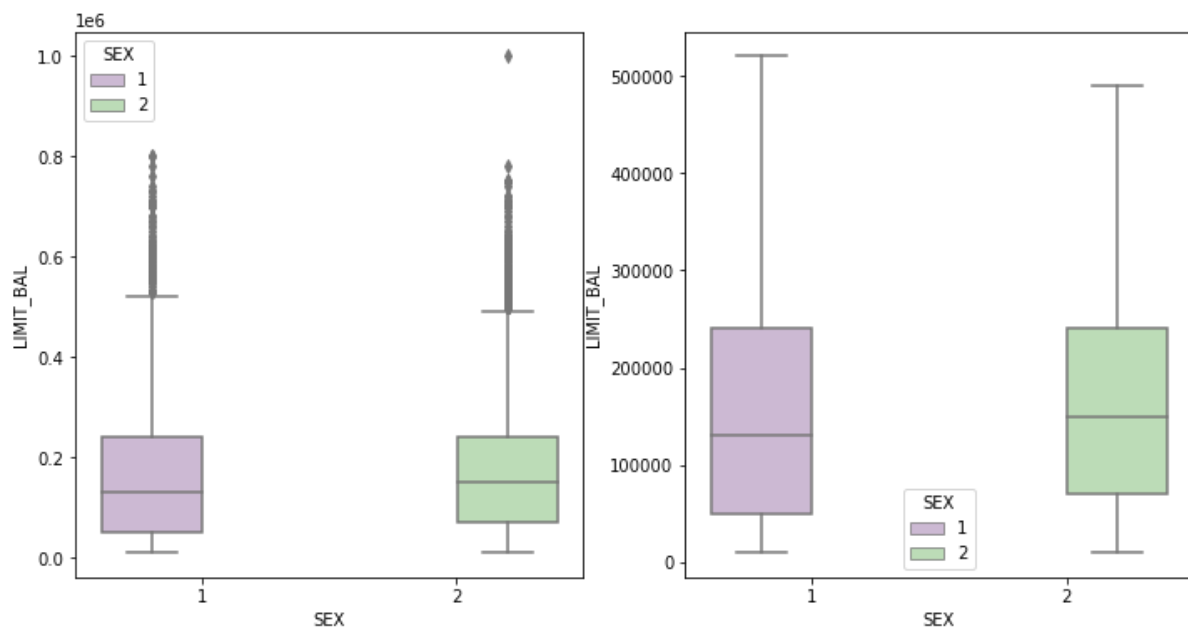


Most of defaults are for credit limits 0-100,000 (and density for this interval is larger for defaults than for non-defaults). Larger defaults number are for the amounts of 50,000, 20,000 and 30,000.

- Visualizing the Credit Limit relationship with respect to Gender

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

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

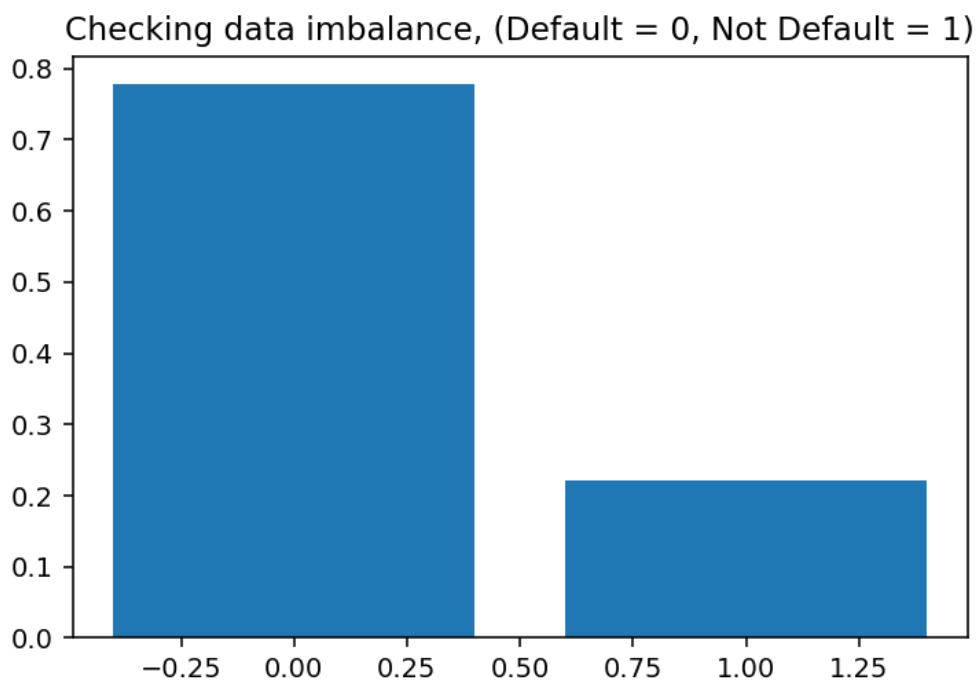
Input	total = data.isnull().sum().sort_values(ascending = False) print(total)	
Output	default payment next month	0
	PAY_6	0
	LIMIT_BAL	0
	SEX	0
	EDUCATION	0
	MARRIAGE	0
	AGE	0
	PAY_0	0
	PAY_2	0
	PAY_3	0
	PAY_4	0
	PAY_5	0
	BILL_AMT1	0
	PAY_AMT6	0
	BILL_AMT2	0
	BILL_AMT3	0
	BILL_AMT4	0
	BILL_AMT5	0
	BILL_AMT6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

	ID	0
	dtype: int64	

- Checking for data imbalance

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

Output

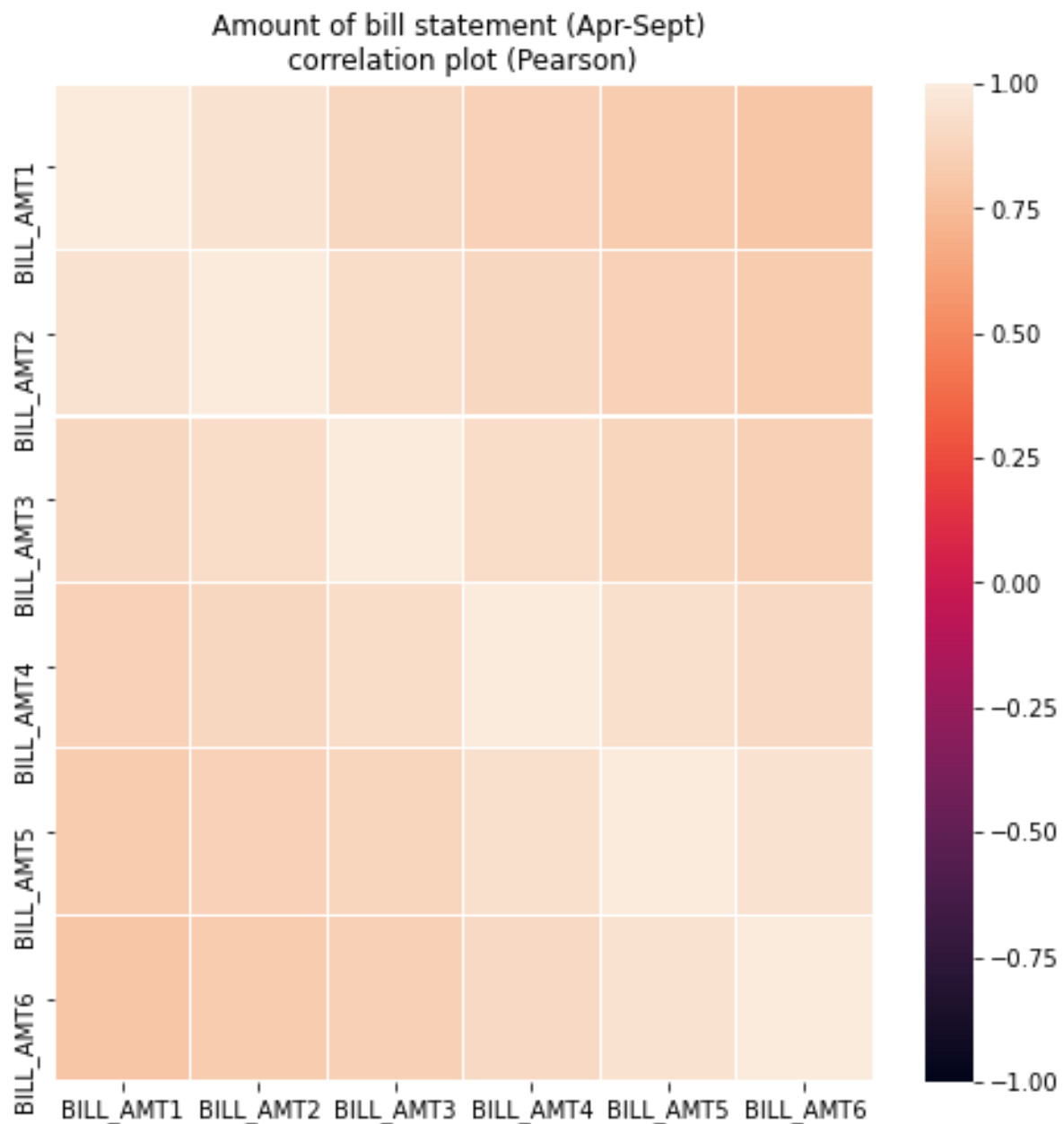


A minimum of 22% of the total clients will default into next month. The data does not have a large imbalance with respect to the target value - Default payment for next month.

5. Feature Correlation Matrix

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

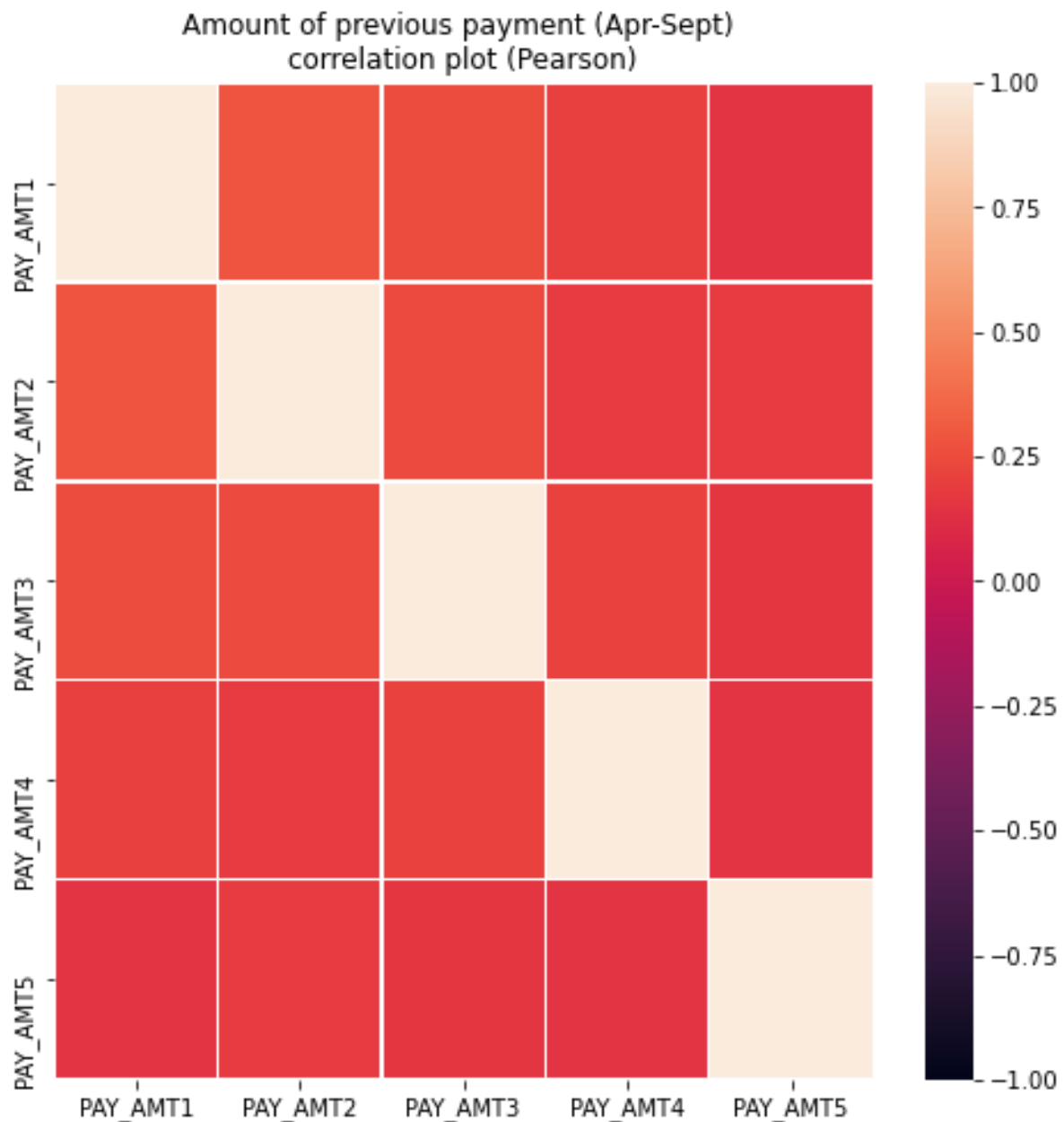
Output



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

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

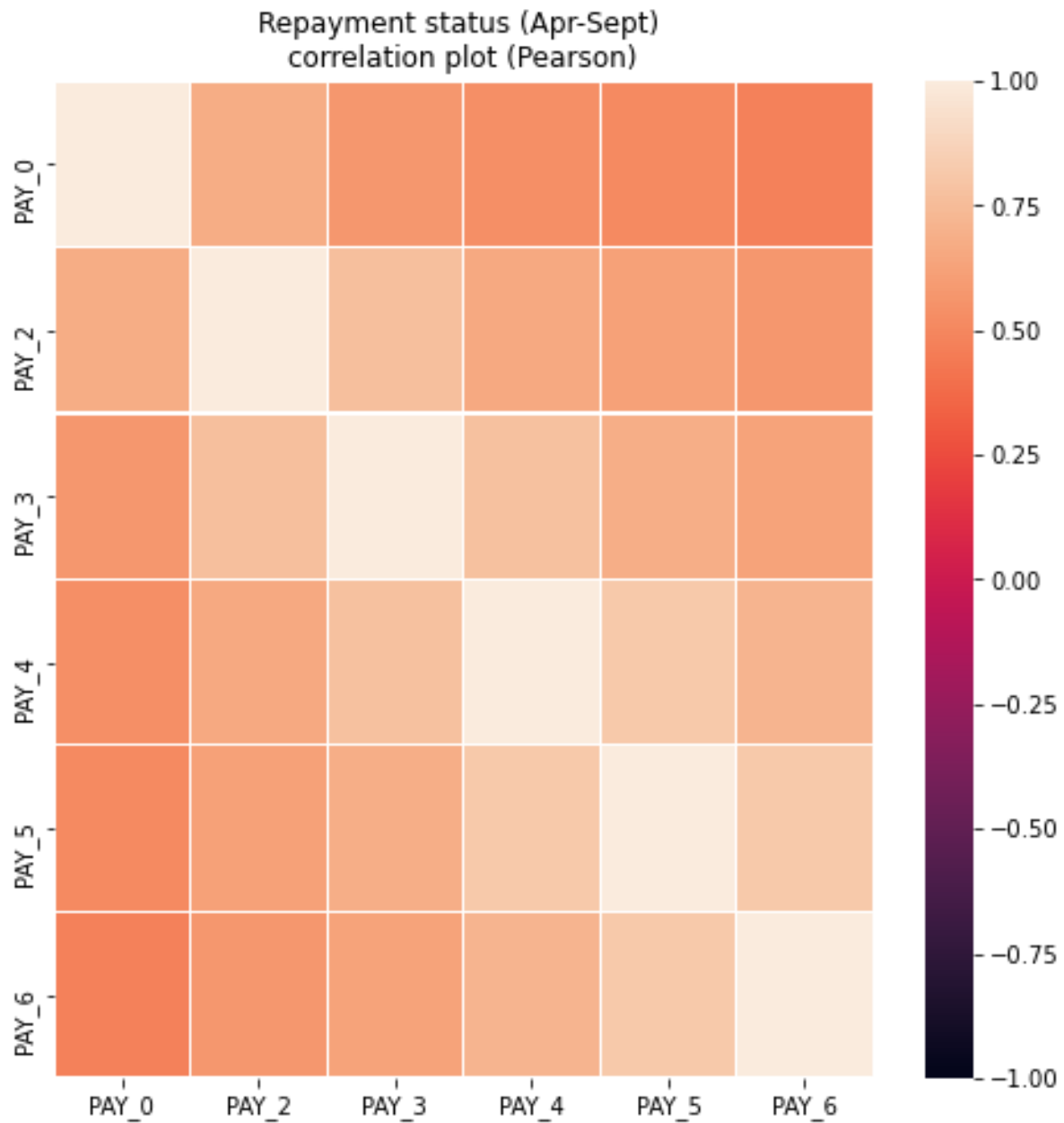
Output



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

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

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:

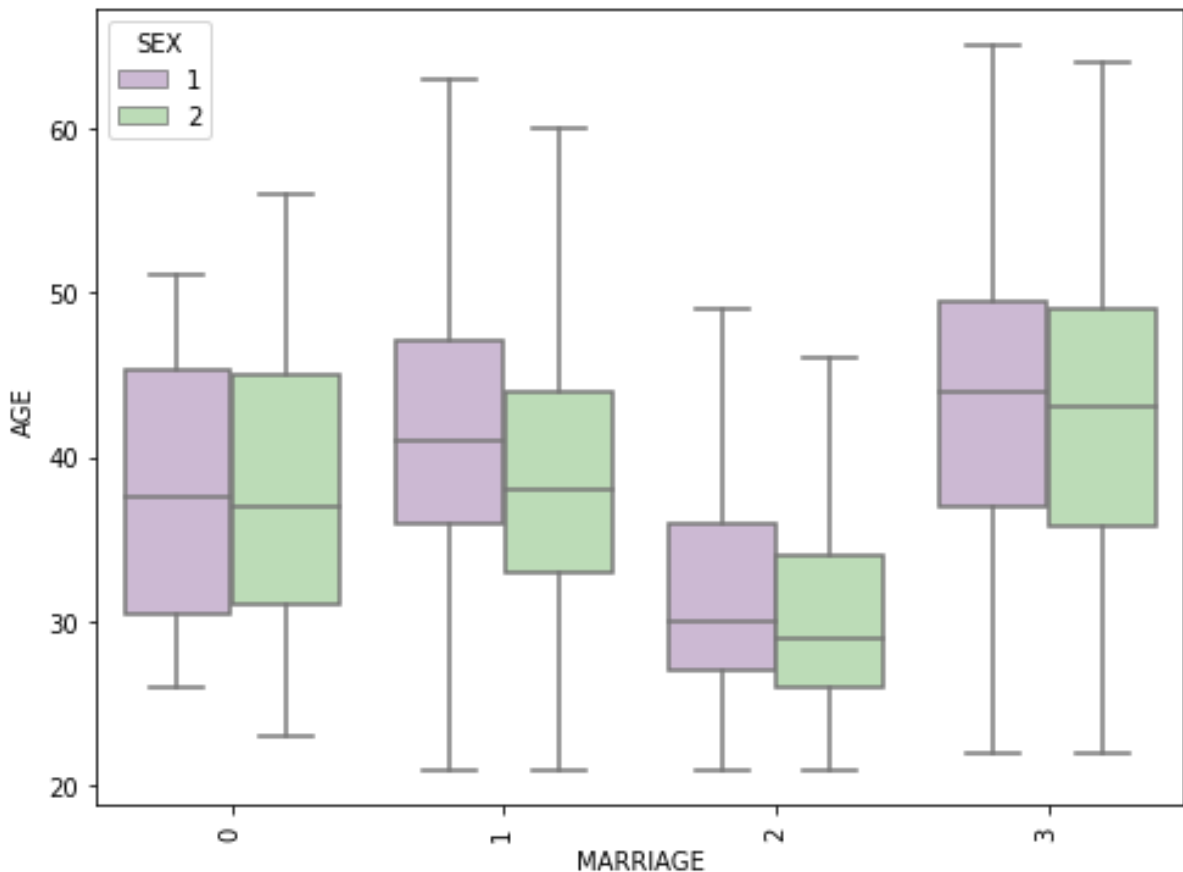
- 0 : unknown (let's consider as others as well)
- 1 : married
- 2 : single
- 3 : others

Sex meaning is:

- 1 : male
- 2 : female

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

Output



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

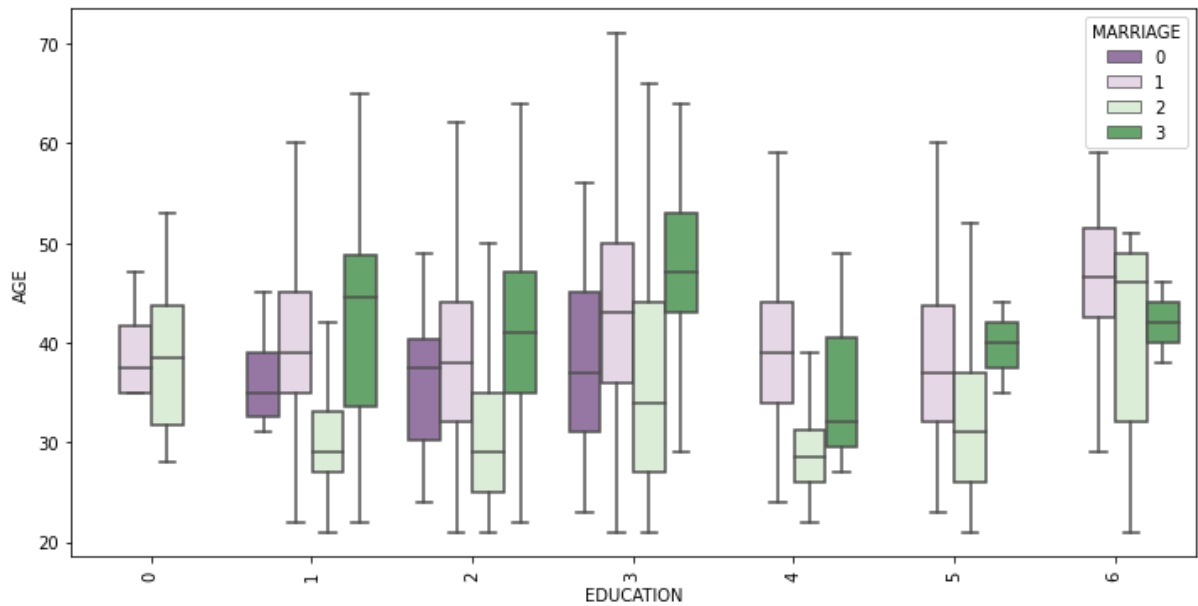
Married males have mean age above married women. Unmarried males have mean value for age above unmarried women as well but closer. Q3 and Q4 values for married man are above corresponding values for married women.

Education status meaning is:

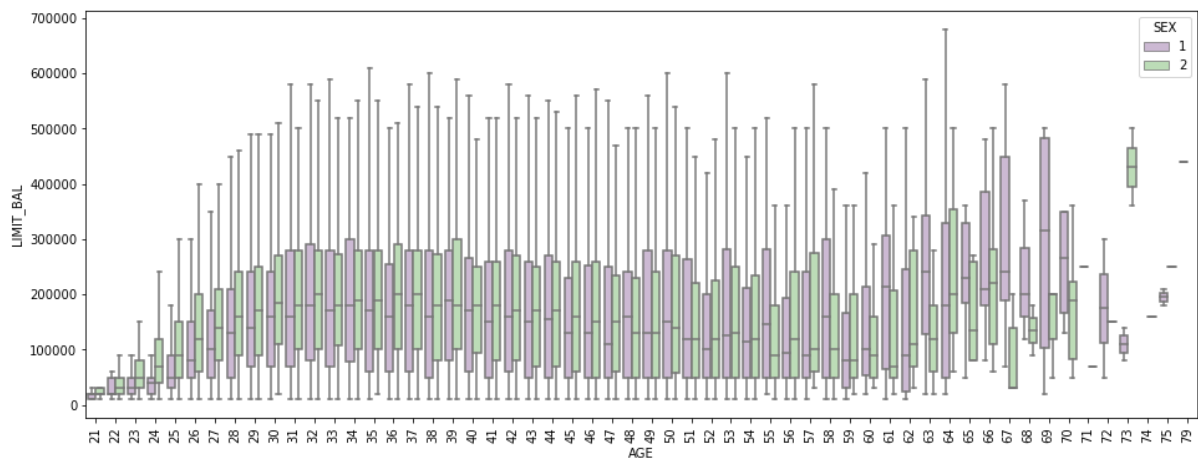
- 1 : graduate school
- 2 : university
- 3 : high school
- 4 : others
- 5 : unknown
- 6 : unknown

Input	<code>boxplot_variation('EDUCATION','AGE', 'MARRIAGE',12)</code>
-------	--

Output



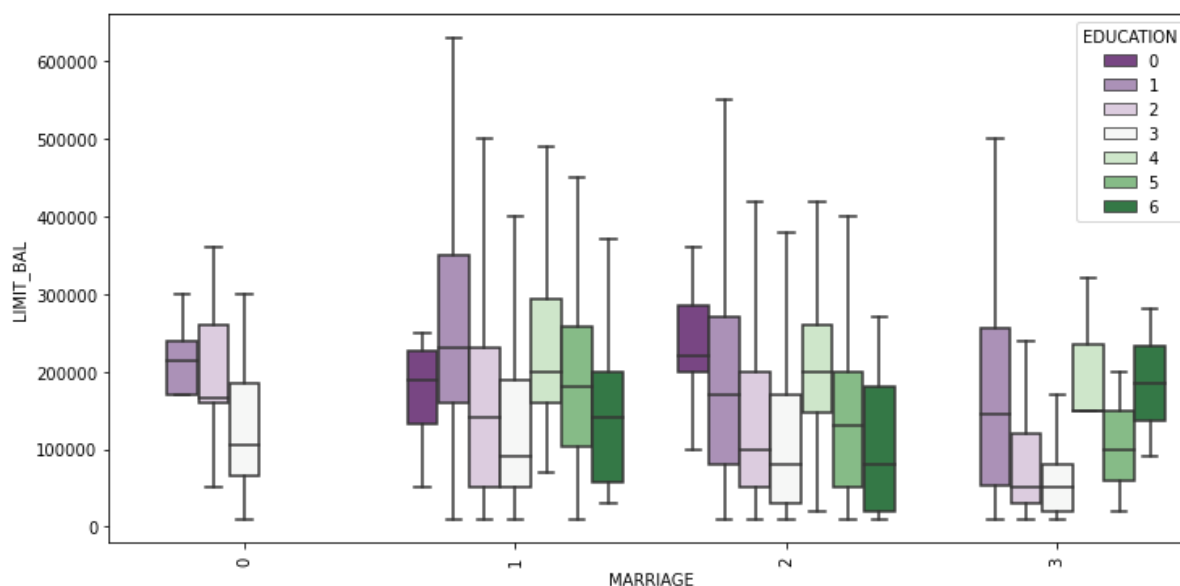
Input	<code>boxplot_variation('AGE','LIMIT_BAL','SEX',16)</code>
Output	



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 around 35 years and then they are oscillating 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	<code>boxplot_variation('MARRIAGE','LIMIT_BAL', 'EDUCATION',12)</code>
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	<pre>target = 'default.payment.next.month' predictors = ['LIMIT_BAL', 'SEX', 'EDUCATION', 'MARRIAGE', 'AGE', 'PAY_0', 'PAY_2', 'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6', 'BILL_AMT1', 'BILL_AMT2', 'BILL_AMT3', 'BILL_AMT4', 'BILL_AMT5', 'BILL_AMT6', 'PAY_AMT1', 'PAY_AMT2', 'PAY_AMT3', 'PAY_AMT4', 'PAY_AMT5', 'PAY_AMT6'] train_df, val_df = train_test_split(data, test_size=0.25, random_state=42, shuffle=True)</pre>
-------	--

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.

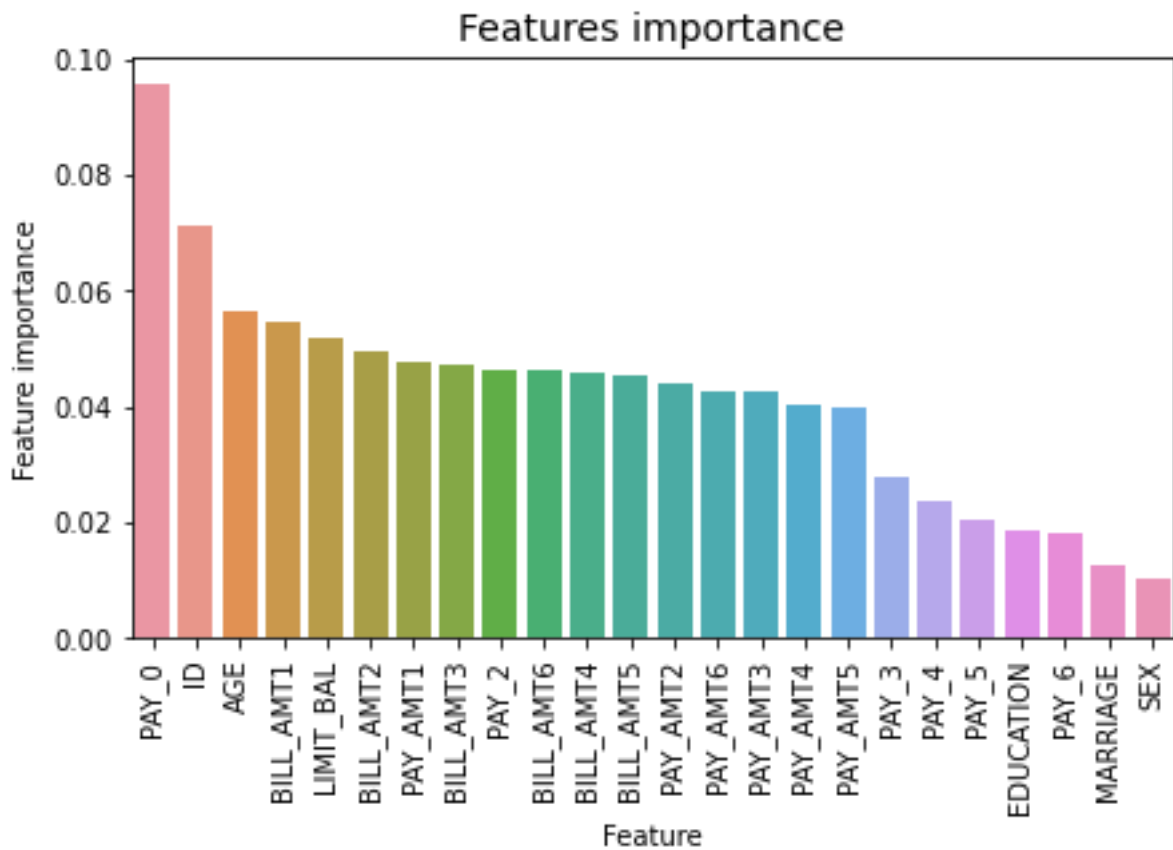
Input	<pre>clf = RandomForestClassifier(n_jobs=-1, random_state=42, criterion='gini', n_estimators=350, verbose=False) predictors = train_df.drop(columns=['default payment next month']).columns.values target = 'default payment next month'</pre>
-------	---

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

- Features Importance

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

Output

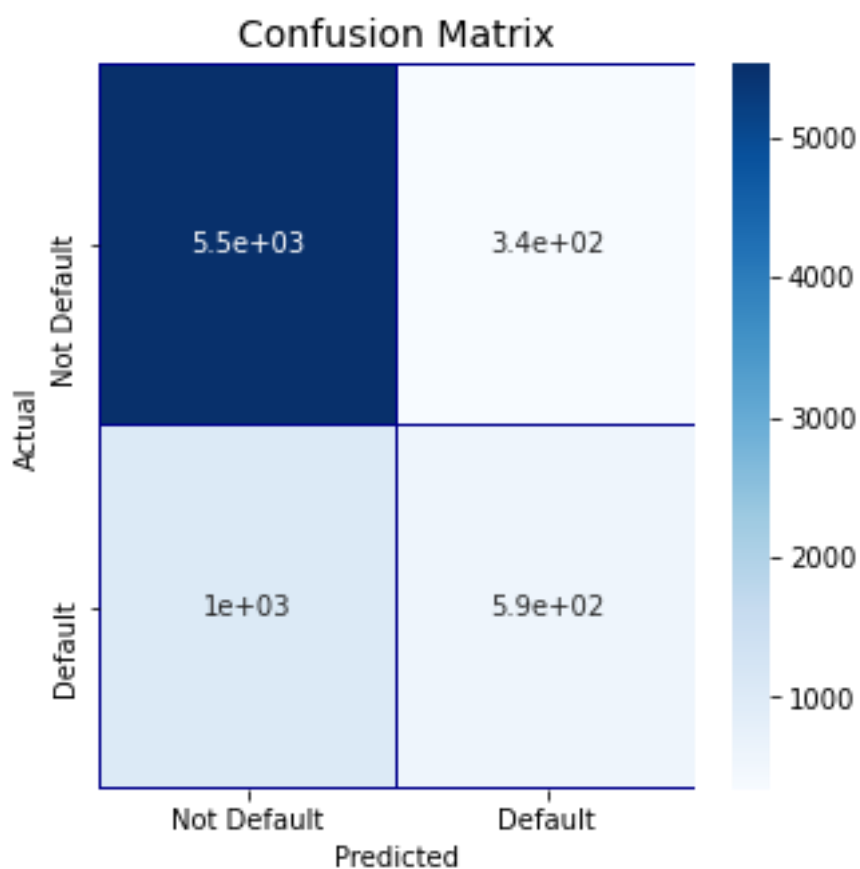


The most important features are PAY_0, AGE, BILL_AMT1, LIMIT_BAL, BILL_AMT2, BILL_AMT3.

- Confusion Matrix

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

Output



9. Random Forest with One Hot Encoder

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

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	1501
22404	26	136736	125651	116684	101581	77741	77264	0	1	0	...	0	4481
23397	32	70122	69080	68530	69753	70111	70212	0	0	0	...	0	2431
25058	49	20678	18956	16172	16898	11236	6944	0	0	0	...	0	1611
2664	36	94228	47635	42361	19574	20295	19439	0	0	1	...	0	2001

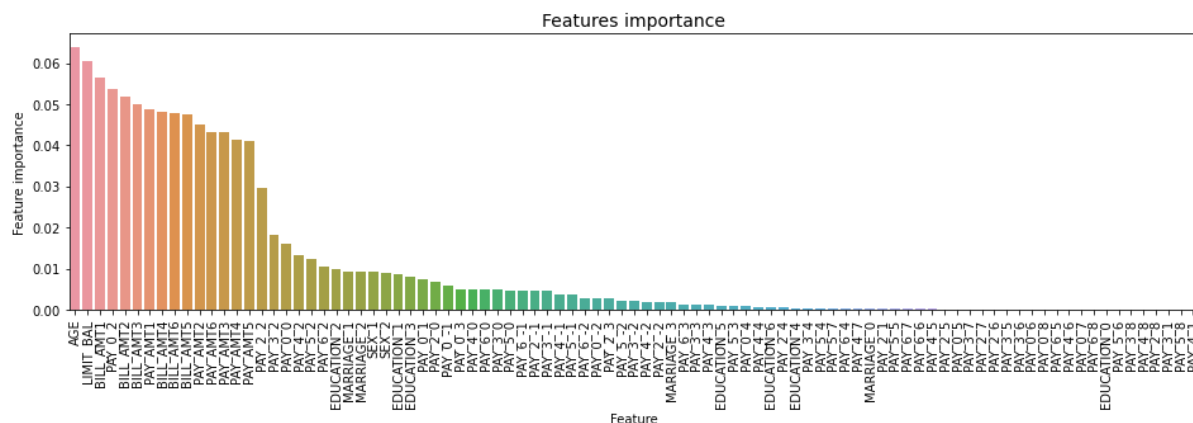
5 rows × 93 columns

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

- Features Importance

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

Output

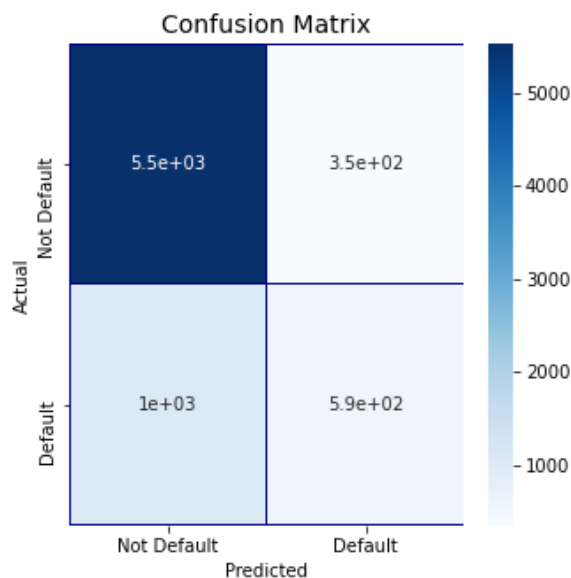


The most important features are AGE, LIMIT_BAL, BILL_AMT1, PAY_0_2, BILL_AMT2, BILL_AMT3.

- Confusion Matrix

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

Output



Output

Input	<code>roc_auc_score(val_df[target].values, preds)</code>
Output	<code>0.6532579949015062</code>

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	<code>_auc_score(val_fa_df[target].values, preds)</code>
Output	<code>0.6507662549156908</code>

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**.