CMP7288 Machine Learning

LOAN PREDICTION BASED ON CUSTOMER BEHAVIOR

Chinedum Vincent Okereafor

MSC Artificial Intelligence

22179676

TABLE OF CONTENT

Link to code: https://colab.research.google.com/drive/1VdagW6gN3-SkbNTDy6eruw7_edWsZWS2?usp=sharing

Link to dataset: https://raw.githubusercontent.com/VincenteCodes/Loan-prediction/main/Training%20Data.csv

Introduction

The two most crucial inquiries in the loan industry are: How risky is the borrower? 2) Given the risk involved, should we lend to the borrower? The answer to the first question would determine the interest rate for the borrower. One of the variables that affects interest rates is borrower riskiness; the more the riskier the borrower, the higher the interest rate. The borrower's loan eligibility can then be determined depending on the interest rate.

Lenders (investors) provide borrowers with loans in exchange for the promise of interest-bearing repayment. Thus, the lender only receives payment (interest) if the borrower pays the obligation in full. However, if the borrower defaults on the loan, the lender will suffer a financial loss.

What a loan is:

A loan is anything that is offered to someone in exchange for future repayment of the loan value plus interest and fees in accordance with an agreement signed before the loan is granted. It can be money, tangible things, property, etc.

A loan may have a fixed value or a range of amounts with a maximum amount that will be provided.

A loan can have a fixed value or a range of amounts with a maximum amount that can be approved, in this instance the credit limit.

A loan can be obtained in a variety of ways, including personal, business, secured, and unsecured.

*Detailed explanation of the loan:

One could define a loan as a debit committed by a person, a group of people, or an organisation.

The person in this situation is referred to as the debtor, and the entity providing the loan is referred to as the creditor or lender. A lender or creditor can also be a single person, a group of people, a financial institution, or any other type of company.

Before a loan is given, an agreement is made between the two parties.

Getting a Loan:

When a person needs money, they apply for a loan, either directly or through a financial institution, and often include the following information: Loan amount, personal information, addresses for both their home and place of employment, and other pertinent details. The financial institution processes the application after it is received, at which point it decides whether the applicant is creditworthy. Creditworthiness is assessed using a variety of factors, including debt-to-income ratio, loan history, and other factors.

If the loan application is approved, a contract between the parties is signed, and the lender gives or transfers the loan amount to the borrower. If the loan application is rejected for any reason(s), the lender explains why.

*

Loans are repaid for a variety of reasons. A borrower might require a loan to make a purchase, pay fees, go on vacation or travel, buy medicine, start a business, do repairs, or make an investment.

A business loan is one that an organisation might obtain to aid in the expansion of its operations.

Before taking out a loan, a person must take a number of aspects into account, one of which is the interest rate. If the borrower takes out a loan with a high interest rate, they will have to pay more money back In comparison to a loan with a lower interest rate, he must make larger monthly payments or repay the debt over a longer period of time.

For instance, if someone loans £5,000 over the course of five years in equal payments at a 4.5% interest rate, they will have to make a monthly payment of £93.22 over the course of that time. For the same period, the person will need to pay £103.79 if the interest rate is 9%, though. To put things in perspective, if someone borrows £10,000 at an interest rate of 6% and decides to pay £200 per month, it will take them 58 months to pay off the debt. The loan will be repaid in 108 months at 20% interest with the same balance and payment schedule.

Customer Behaviors

Customer behaviour refers to a person's buying habits, including the frequency of purchases, societal trends, and external factors that affect a person's decision to make a purchase. Businesses study customer behaviour to better understand their target market and create products and service offerings that are more appealing.

Customer behaviour is more concerned with how people shop at a store than with who makes purchases there. It examines some elements including buying habits, product preferences, and how marketing, sales, and provided services are seen. Businesses can communicate with clients in a more fruitful and enjoyable way by being aware of these details.

Customer Behavior Analysis:

Analyzing customer behaviour involves both qualitative and quantitative observations of how customers engage with a company. This is accomplished by creating several consumer segments and grouping clients into these segments according to their common interests. After then, keep an eye on each segment at the appropriate point in the customer journey to watch how the various personas interact with the business.

The consumer behaviour analysis aids in providing information about the factors that affect the company's target markets and the motivations, objectives, and decision-making processes that customers take into account while travelling. Understanding how customers feel about the business and whether their view matches their basic principles is also helpful.

Domain Description:

The finance sector is immensely large and vital to society because no society can run properly without it.

The importance of credit institutions cannot be overstated as both individuals and businesses require loans and credit to operate and live comfortably.

Credit is crucial to the growth and development of a country because it prevents people from having to live in squalor due to a lack of resources.

The phrase "Finance Domain" is typically used to describe the knowledge and professions related to the finance sector or financial services. There could be some misunderstanding as to what we mean when we use the phrase "Finance Domain." Financial institutions exist in addition to the function known as finance, which is shared by all business firms across all industries. Possibilities for careers within the financial services sector or with financial institutions are made available by knowledge in the finance domain.

There could be some misunderstanding as to what we mean when we use the phrase "Finance Domain."

On the one hand, all business enterprises and industries share a function called finance, and we have covered these financial operations under functional skills. The financial services sector exists in contrast. Then, there is a phrase that covers all types of financial establishments, including banks, brokerage firms, insurance companies, etc. Possibilities for careers within the financial services sector or with financial institutions are made available by knowledge in the finance domain.

The Finance Department

Many different processes, activities, and functions are included in the finance function. It includes budgetary and financial management, risk and return management, cash flow and cash management, financial management, risk and governance, and numerous other related tasks.

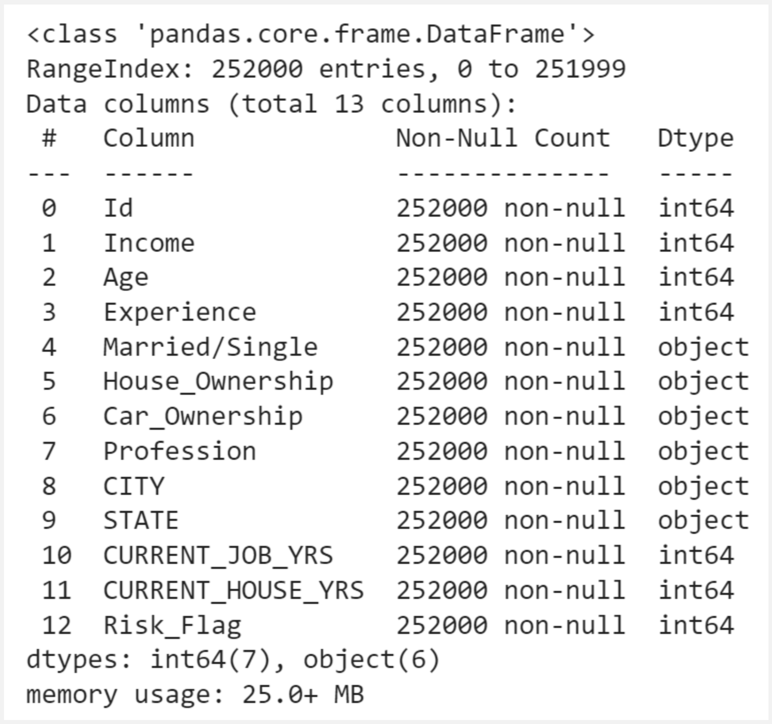
Problems Definition

A lending institution seeks to identify potential defaulters for its consumer loan products. Based on their observations, they have information about previous client behaviour. As a result, they want to know whether a new customer has a high or low chance of defaulting.

Dataset Description:

The dataset was obtained from Kaggle: https://www.kaggle.com/datasets/subhamjain/loan-prediction-based-on-customer-behavior/code?resource=download

There are 12 variables in this dataset, and all of their values were supplied at the time of the loan application.

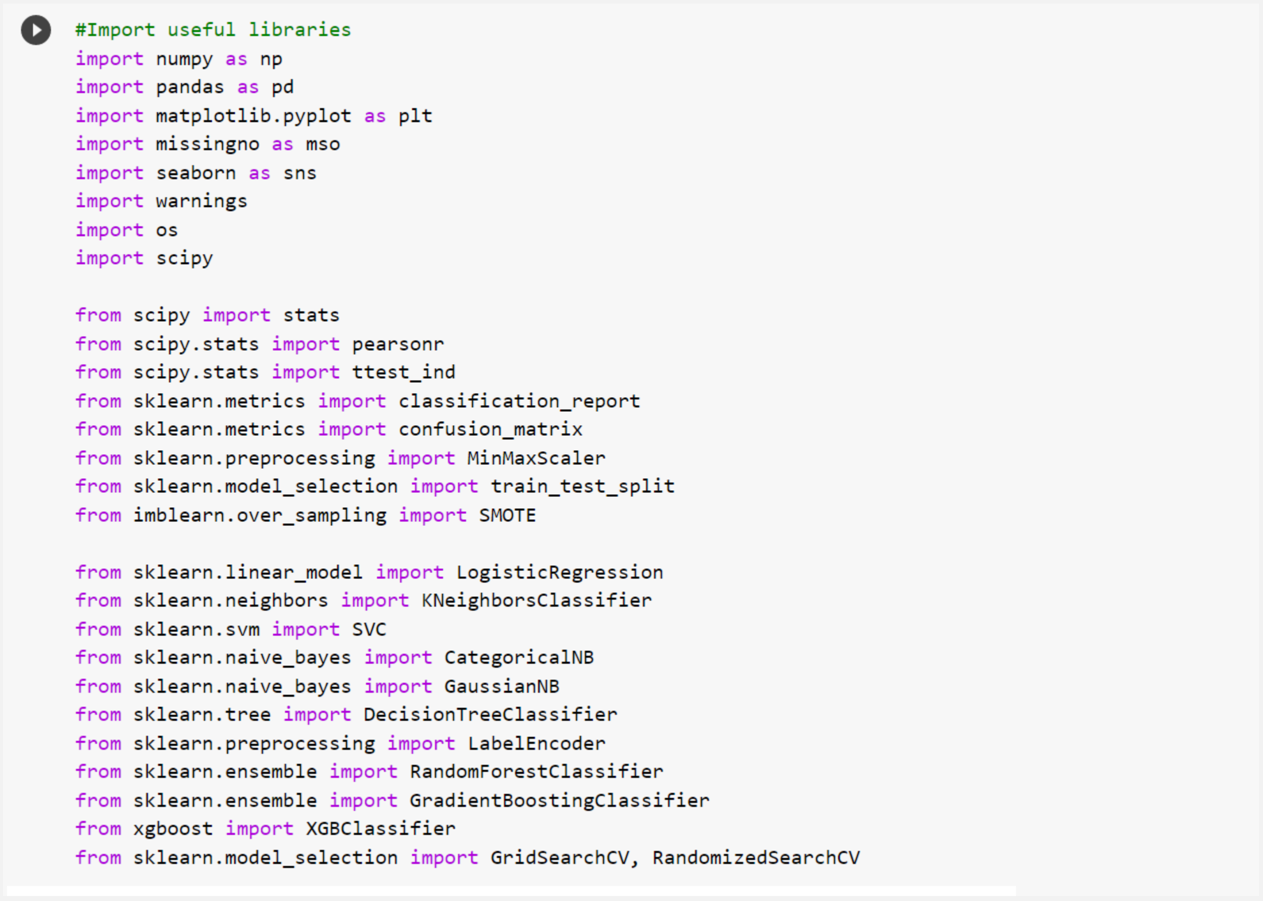


```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 252000 entries, 0 to 251999
Data columns (total 13 columns):
 #   Column             Non-Null Count   Dtype 
---  ------             --------------   ----- 
 0   Id                 252000 non-null  int64 
 1   Income             252000 non-null  int64 
 2   Age                252000 non-null  int64 
 3   Experience         252000 non-null  int64 
 4   Married/Single     252000 non-null  object
 5   House_Ownership    252000 non-null  object
 6   Car_Ownership      252000 non-null  object
 7   Profession         252000 non-null  object
 8   CITY               252000 non-null  object
 9   STATE              252000 non-null  object
 10  CURRENT_JOB_YRS    252000 non-null  int64 
 11  CURRENT_HOUSE_YRS  252000 non-null  int64 
 12  Risk_Flag          252000 non-null  int64 
dtypes: int64(7), object(6)
memory usage: 25.0+ MB
```

The risk flag specifies whether or not the consumer has ever defaulted on a loan. The dataset's entire set of values was supplied at the time the loan application was submitted.

Dataset Pre-processing

The first step is to import the libraries we will use for this research.

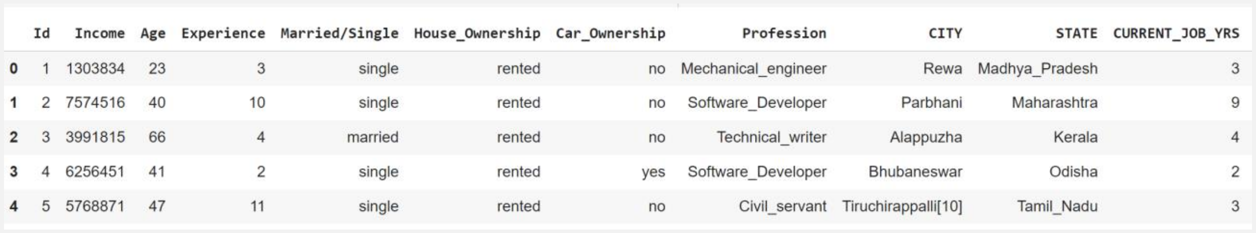


```python
#Import useful libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import missingno as mso
import seaborn as sns
import warnings
import os
import scipy

from scipy import stats
from scipy.stats import pearsonr
from scipy.stats import ttest_ind
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE

from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import CategoricalNB
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
```

Reading the csv file from local or the online is the next step after importing the crucial libraries. In this instance, the csv is being read from my github website.
We use the following function to read CSV files:

[in]
```python
df = pd.read_csv("https://raw.githubusercontent.com/VincenteCodes/Loan-prediction/main/Training%20Data.csv")
df.head()
```

The output of the aforementioned input is as follows:



	Id	Income	Age	Experience	Married/Single	House_Ownership	Car_Ownership	Profession	CITY	STATE	CURRENT_JOB_YRS
0	1	1303834	23	3	single	rented	no	Mechanical_engineer	Rewa	Madhya_Pradesh	3
1	2	7574516	40	10	single	rented	no	Software_Developer	Parbhani	Maharashtra	9
2	3	3991815	66	4	married	rented	no	Technical_writer	Alappuzha	Kerala	4
3	4	6256451	41	2	single	rented	yes	Software_Developer	Bhubaneswar	Odisha	2
4	5	5768871	47	11	single	rented	no	Civil_servant	Tiruchirappalli[10]	Tamil_Nadu	3

The next step is to perform exploratory data analysis (EDA) on the data. During this process, we look for null values and attempt to visualise the data using a variety of plots and graphs. Being familiar with our dataset and alert to any missing numbers or discrepancies in our data is the key to this approach.

EDA also enables us to assess a dataset's usability.

Exploratory data analysis involves the following steps:

• Data collection: This is the procedure by which we load our dataset into our system, in this example, our Google Colab workspace.

• Data cleaning involves taking out all unnecessary variables and values from our dataset and fixing any abnormalities. These anomalies in our dataset have the potential to significantly skew the data and have an impact on the outcomes.

Removing missing values, outliers, and pointless rows and columns, as well as reformatting and re-indexing our data, are further aspects of data cleaning.

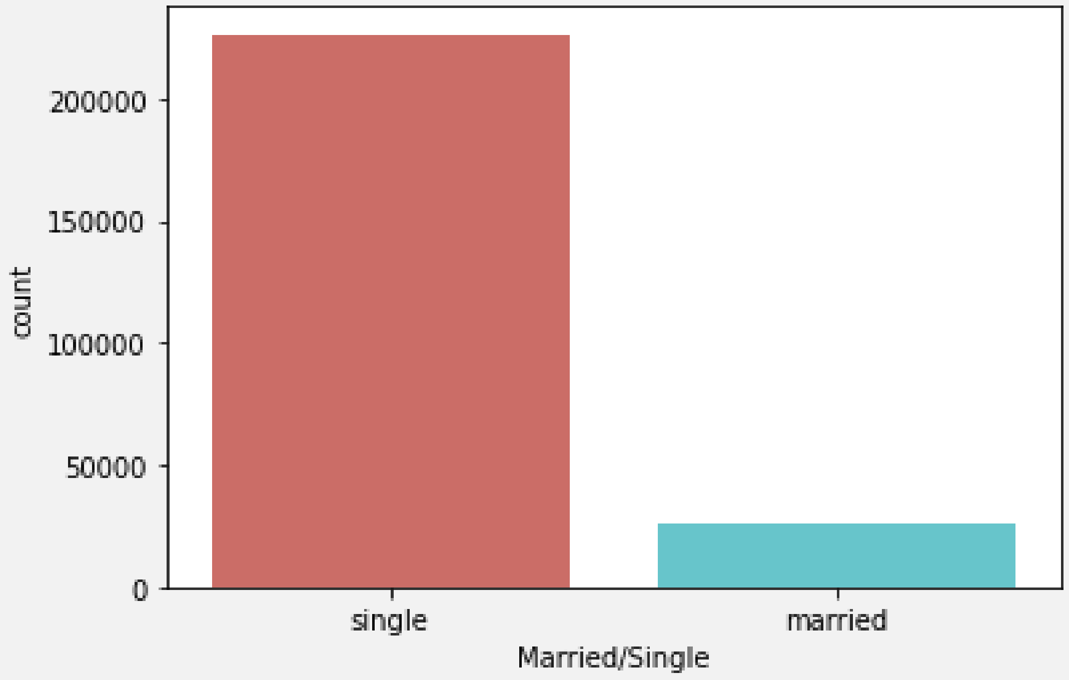
• We investigate data with only one variable using a method referred to as "univariate analysis." A variable in our dataset denotes a single feature or column. We can do this either visually or non-graphically by finding the precise mathematical values in the data. These methods include the histogram, bar chart, box plot, and pie chart, among others.

• Bivariate analysis: Similar to the preceding instance, bivariate analysis involves considering two variables and comparing them to one another. In order to better understand both variables, we look for commonalities.

• Multivariate analysis is the term for the analysis of more than two variables in which we look at how the variables differ from one another and how well they compare.

```
[in] sns.countplot(x="Married/Single", data=df, palette="hls")
     plt.show()
```

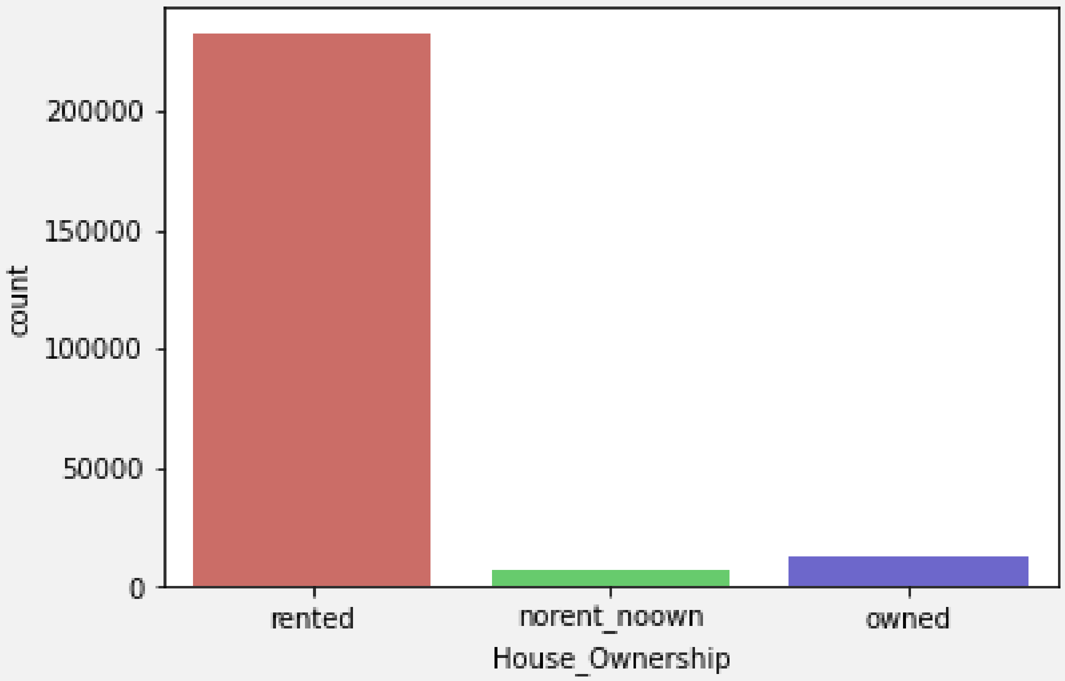
The contrast between married and single candidates, where the majority of applicants are single, is shown in the code above. The following is the code's output:



The similar visualisation can be done for other criteria, such as the applicant's housing type to determine the percentage of home owners, applicants with rented flats, and candidates who neither rent nor own their homes.

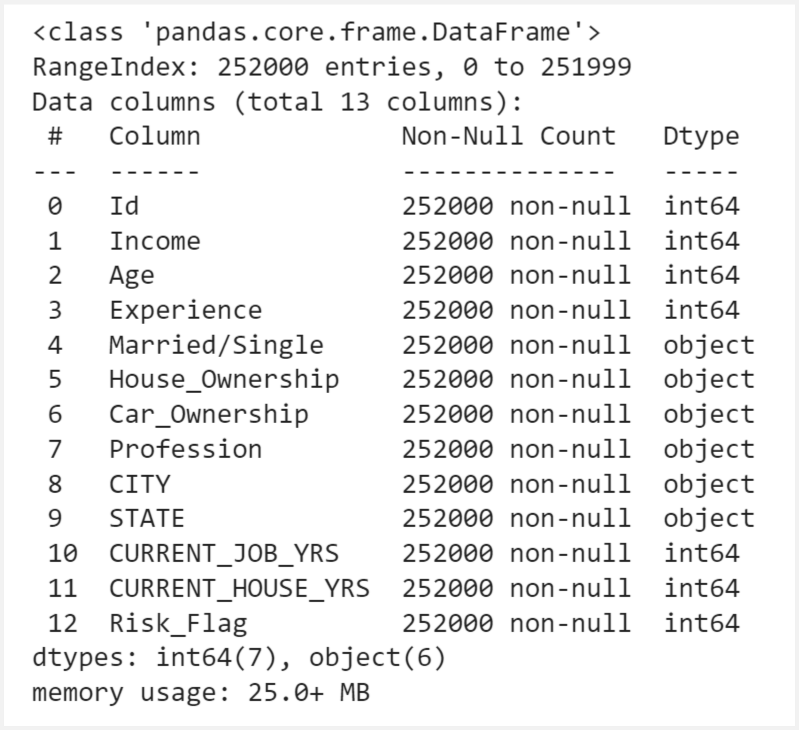
```
[in]sns.countplot(x="House_Ownership", data=df, palette="hls")
    plt.show()
```

[out]



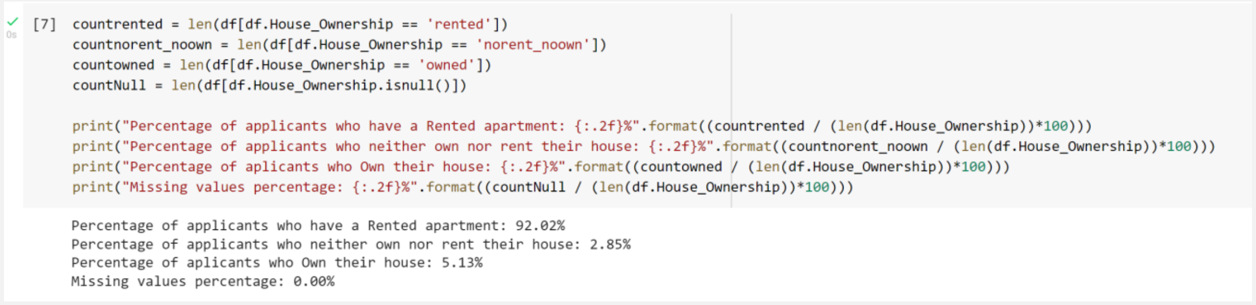
To avoid future problems with our model, we must be on the lookout for null values and either replace them or eliminate them throughout the data cleaning phase. The code below displays the information of the dataframe in summary and allows us to check for null values. Output is represented by [out] and input is represented as [in]:

```
[in] df.info()
[out]
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 252000 entries, 0 to 251999
Data columns (total 13 columns):
 #   Column             Non-Null Count   Dtype 
---  ------             --------------   ----- 
 0   Id                 252000 non-null  int64 
 1   Income             252000 non-null  int64 
 2   Age                252000 non-null  int64 
 3   Experience         252000 non-null  int64 
 4   Married/Single     252000 non-null  object
 5   House_Ownership    252000 non-null  object
 6   Car_Ownership      252000 non-null  object
 7   Profession         252000 non-null  object
 8   CITY               252000 non-null  object
 9   STATE              252000 non-null  object
 10  CURRENT_JOB_YRS    252000 non-null  int64 
 11  CURRENT_HOUSE_YRS  252000 non-null  int64 
 12  Risk_Flag          252000 non-null  int64 
dtypes: int64(7), object(6)
memory usage: 25.0+ MB
```

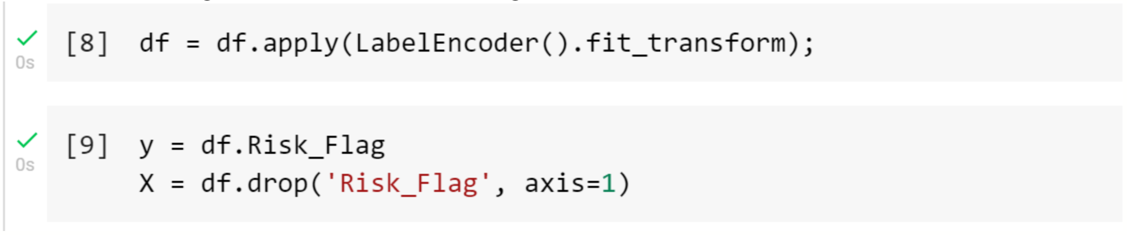
There are no null values, 13 distinct variables, and 252,000 rows or instances, as can be seen in the output above.



```
[7] countrented = len(df[df.House_Ownership == 'rented'])
    countnorent_noown = len(df[df.House_Ownership == 'norent_noown'])
    countowned = len(df[df.House_Ownership == 'owned'])
    countNull = len(df[df.House_Ownership.isnull()])

    print("Percentage of applicants who have a Rented apartment: {:.2f}%".format((countrented / (len(df.House_Ownership))*100)))
    print("Percentage of applicants who neither own nor rent their house: {:.2f}%".format((countnorent_noown / (len(df.House_Ownership))*100)))
    print("Percentage of aplicants who Own their house: {:.2f}%".format((countowned / (len(df.House_Ownership))*100)))
    print("Missing values percentage: {:.2f}%".format((countNull / (len(df.House_Ownership))*100)))

    Percentage of applicants who have a Rented apartment: 92.02%
    Percentage of applicants who neither own nor rent their house: 2.85%
    Percentage of aplicants who Own their house: 5.13%
    Missing values percentage: 0.00%
```

According to the aforementioned finding, there are more applicants with rented apartments than candidates who own their homes and those who neither rent nor own their homes. Additionally, it is obvious that this column contains no missing values.

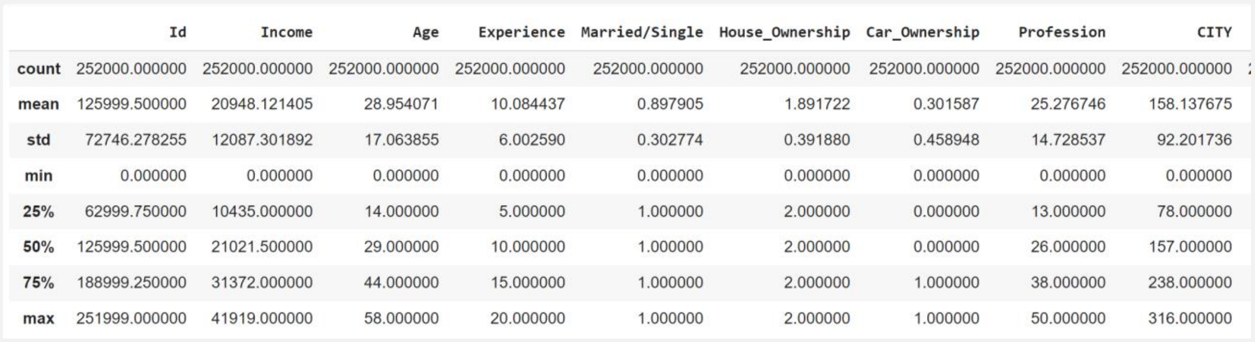
Label encoding in machine learning is the process of transforming a categorical variable's values into integer values. The following code below will allow us to execute that:



```
[8] df = df.apply(LabelEncoder().fit_transform);

[9] y = df.Risk_Flag
    X = df.drop('Risk_Flag', axis=1)
```

```
#Brief desciption of the data
[in]df.describe()
```

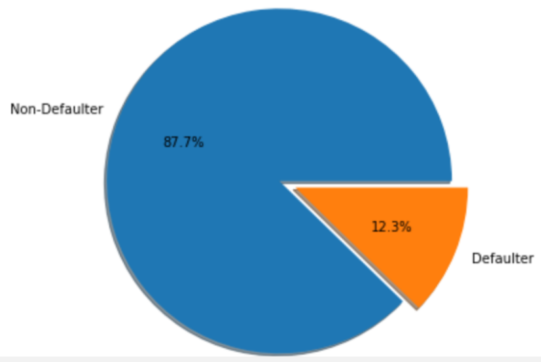
[out]



	Id	Income	Age	Experience	Married/Single	House_Ownership	Car_Ownership	Profession	CITY
count	252000.000000	252000.000000	252000.000000	252000.000000	252000.000000	252000.000000	252000.000000	252000.000000	252000.000000
mean	125999.500000	20948.121405	28.954071	10.084437	0.897905	1.891722	0.301587	25.276746	158.137675
std	72746.278255	12087.301892	17.063855	6.002590	0.302774	0.391880	0.458948	14.728537	92.201736
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	62999.750000	10435.000000	14.000000	5.000000	1.000000	2.000000	0.000000	13.000000	78.000000
50%	125999.500000	21021.500000	29.000000	10.000000	1.000000	2.000000	0.000000	26.000000	157.000000
75%	188999.250000	31372.000000	44.000000	15.000000	1.000000	2.000000	1.000000	38.000000	238.000000
max	251999.000000	41919.000000	58.000000	20.000000	1.000000	2.000000	1.000000	50.000000	316.000000

The next step is to perform more data visualisations. For this, we can utilise pictorial plots, and we will experiment with as many as we can in order to have a high level of trust in our dataset. The input and output below demonstrate the different plots we used.

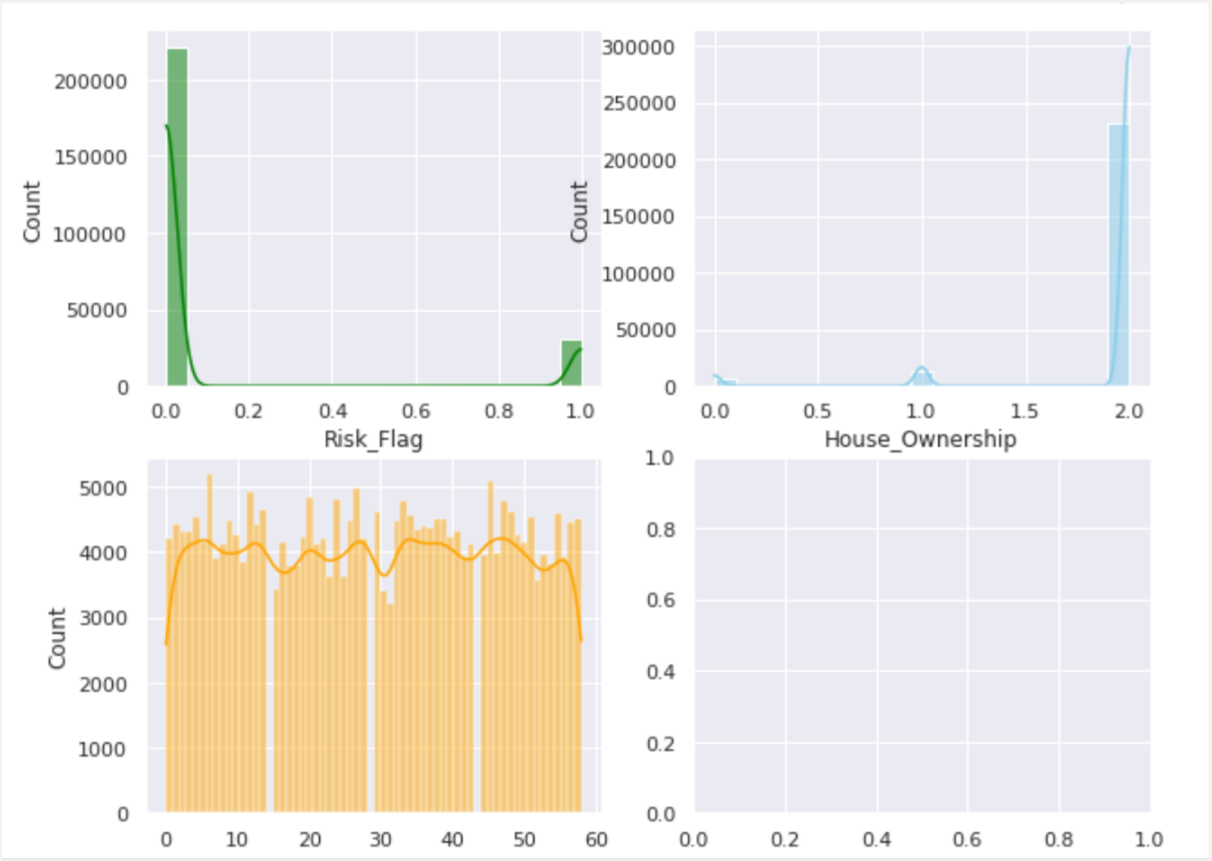
```
#check for percentage of defualters in the data
r = df.groupby('Risk_Flag')['Risk_Flag'].count()

plt.pie(r, explode=[0.05, 0.1], labels=['Non-Defaulter', 'Defaulter'], radius=1.5, autopct='%1.1f%%', shadow=True);
```



A bigger number of prospective non-defaulters (87.7%) are represented in the pie chart above than potential defaulters (12.3%).

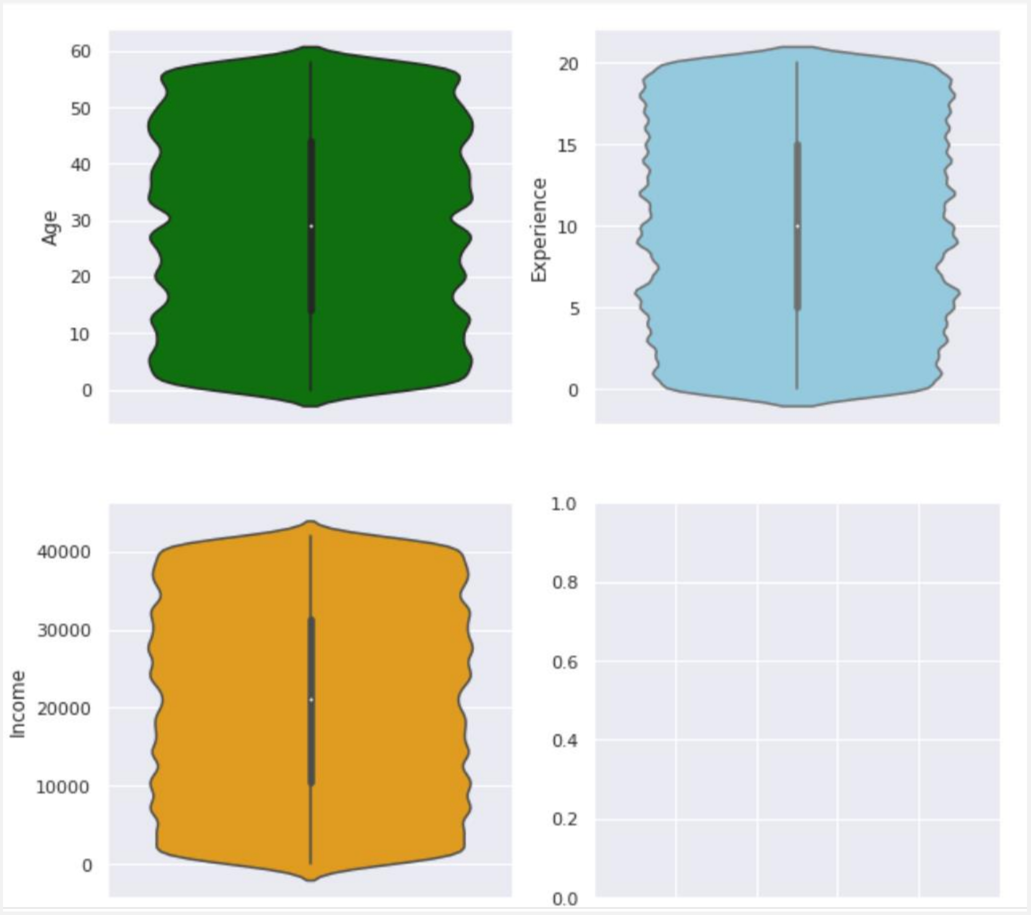
Histogram Distribution

```
sns.set(style="darkgrid")
fig, axs = plt.subplots(2, 2, figsize=(10, 8))

sns.histplot(data=df, x="Risk_Flag", kde=True, ax=axs[0, 0], color='green')
sns.histplot(data=df, x="House_Ownership", kde=True, ax=axs[0, 1], color='skyblue')
sns.histplot(data=df, x="Age", kde=True, ax=axs[1, 0], color='orange');
```



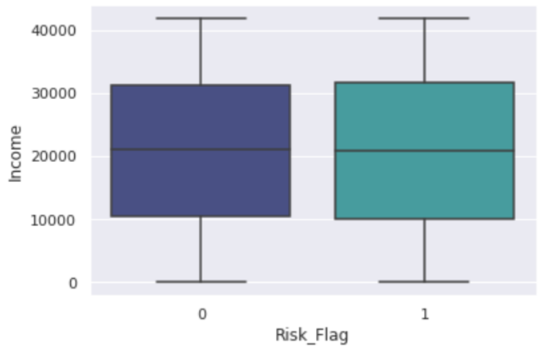
Violin Plot 🎻

```
sns.set(style="darkgrid")
fig, axs1 = plt.subplots(2, 2, figsize=(10, 10))

sns.violinplot(data=df, y="Age", ax=axs1[0, 0], color='green')
sns.violinplot(data=df, y="Experience", ax=axs1[0, 1], color='skyblue')
sns.violinplot(data=df, y="Income", ax=axs1[1, 0], color='orange');
```

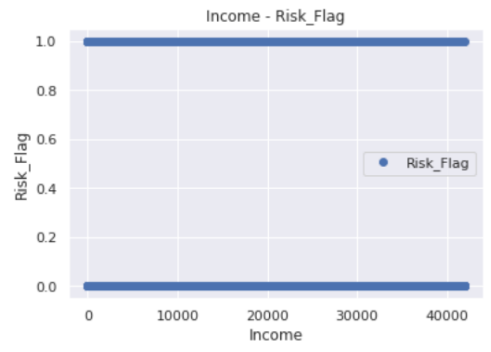


Outliers exist in the distribution of applicant age and applicant income, which is favourably biassed (can be seen from both histogram and violin plot).

```
[15] sns.boxplot(x="Risk_Flag", y="Income", data=df, palette="mako");
```



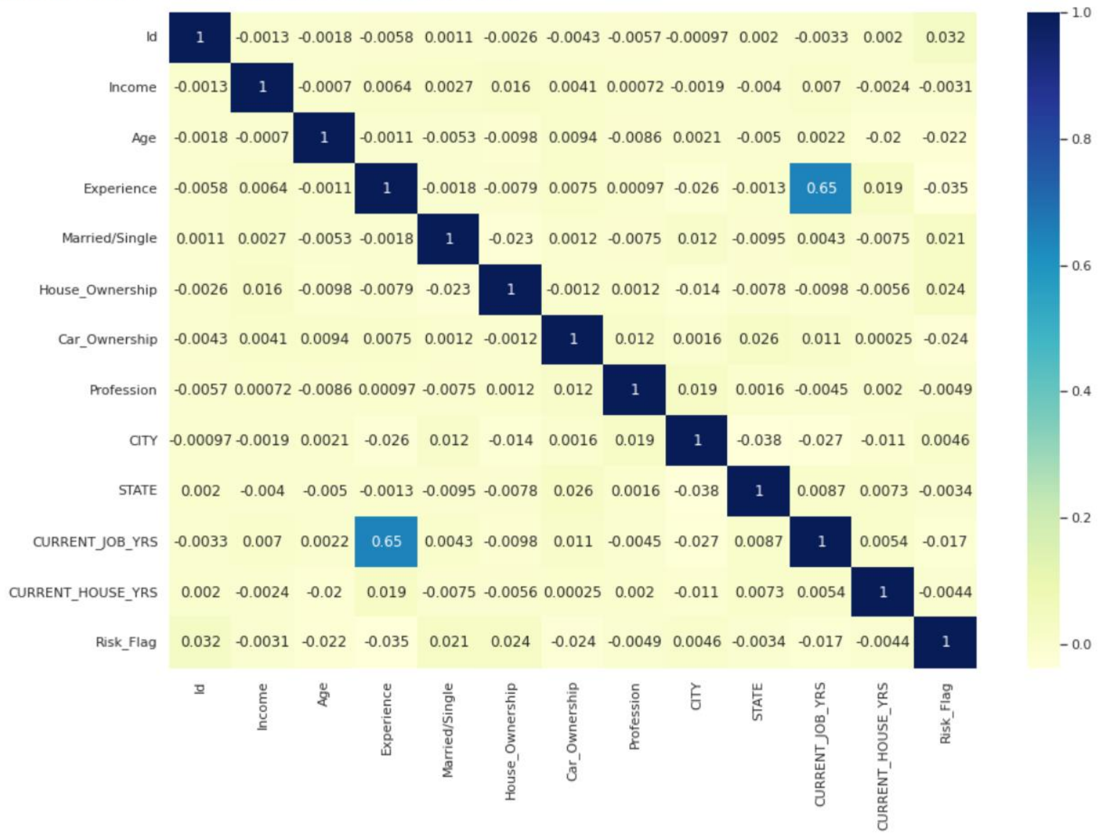
```
df.plot(x='Income', y='Risk_Flag', style='o')
plt.title('Income - Risk_Flag')
plt.xlabel('Income')
plt.ylabel('Risk_Flag')
plt.show()
print('Pearson correlation:', df['Income'].corr(df['Risk_Flag']))
print('T Test and P value: \n', stats.ttest_ind(df['Income'], df['Risk_Flag']))
```



```
Pearson correlation: -0.0031071149785928255
T Test and P value:
 Ttest_indResult(statistic=869.9883423466636, pvalue=0.0)
```

Heatmap

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9ca49891c0>
```

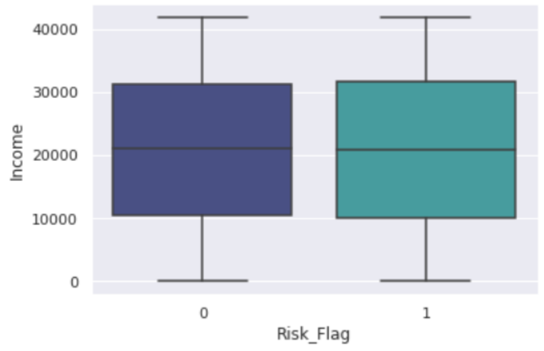


	Id	Income	Age	Experience	Married/Single	House_Ownership	Car_Ownership	Profession	CITY	STATE	CURRENT_JOB_YRS	CURRENT_HOUSE_YRS	Risk_Flag
Id	1	-0.0013	-0.0018	-0.0058	0.0011	-0.0026	-0.0043	-0.0057	-0.00097	0.002	-0.0033	0.002	0.032
Income	-0.0013	1	-0.0007	0.0064	0.0027	0.016	0.0041	0.00072	-0.0019	-0.004	0.007	-0.0024	-0.0031
Age	-0.0018	-0.0007	1	-0.0011	-0.0053	-0.0098	0.0094	-0.0086	0.0021	-0.005	0.0022	-0.02	-0.022
Experience	-0.0058	0.0064	-0.0011	1	-0.0018	-0.0079	0.0075	0.00097	-0.026	-0.0013	0.65	0.019	-0.035
Married/Single	0.0011	0.0027	-0.0053	-0.0018	1	-0.023	0.0012	-0.0075	0.012	-0.0095	0.0043	-0.0075	0.021
House_Ownership	-0.0026	0.016	-0.0098	-0.0079	-0.023	1	-0.0012	0.0012	-0.014	-0.0078	-0.0098	-0.0056	0.024
Car_Ownership	-0.0043	0.0041	0.0094	0.0075	0.0012	-0.0012	1	0.012	0.0016	0.026	0.011	0.00025	-0.024
Profession	-0.0057	0.00072	-0.0086	0.00097	-0.0075	0.0012	0.012	1	0.019	0.0016	-0.0045	0.002	-0.0049
CITY	-0.00097	-0.0019	0.0021	-0.026	0.012	-0.014	0.0016	0.019	1	-0.038	-0.027	-0.011	0.0046
STATE	0.002	-0.004	-0.005	-0.0013	-0.0095	-0.0078	0.026	0.0016	-0.038	1	0.0087	0.0073	-0.0034
CURRENT_JOB_YRS	-0.0033	0.007	0.0022	0.65	0.0043	-0.0098	0.011	-0.0045	-0.027	0.0087	1	0.0054	-0.017
CURRENT_HOUSE_YRS	0.002	-0.0024	-0.02	0.019	-0.0075	-0.0056	0.00025	0.002	-0.011	0.0073	0.0054	1	-0.0044
Risk_Flag	0.032	-0.0031	-0.022	-0.035	0.021	0.024	-0.024	-0.0049	0.0046	-0.0034	-0.017	-0.0044	1

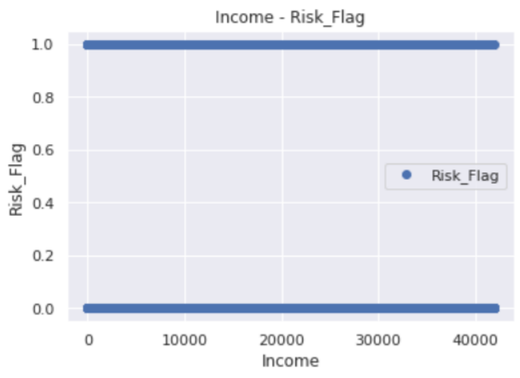
```
[14] #lets check for correlations between variables
     plt.figure(figsize=(15,10))
     sns.heatmap(df.corr(), annot=True, cmap="YlGnBu")
```

Only Experience and CURRENT JOB YRS appear to be slightly connected from the plot above.

```
[15] sns.boxplot(x="Risk_Flag", y="Income", data=df, palette="mako");
```

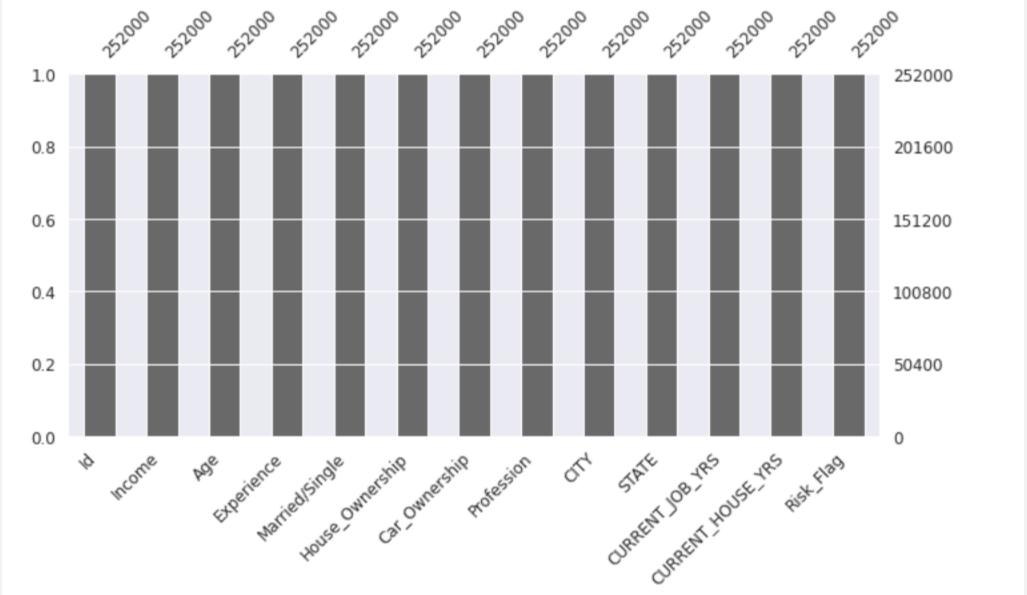


```
df.plot(x='Income', y='Risk_Flag', style='o')
plt.title('Income - Risk_Flag')
plt.xlabel('Income')
plt.ylabel('Risk_Flag')
plt.show()
print('Pearson correlation:', df['Income'].corr(df['Risk_Flag']))
print('T Test and P value: \n', stats.ttest_ind(df['Income'], df['Risk_Flag']))
```



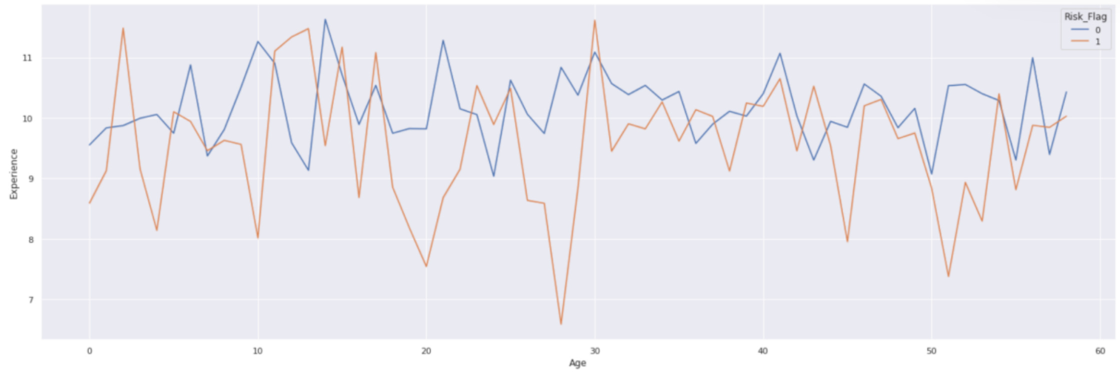
```
Pearson correlation: -0.0031071149785928255
T Test and P value:
 Ttest_indResult(statistic=869.9883423466636, pvalue=0.0)
```

```
plt.figure(figsize = (24, 5))
axz = plt.subplot(1,2,2)
mso.bar(df, ax = axz, fontsize = 12);
```

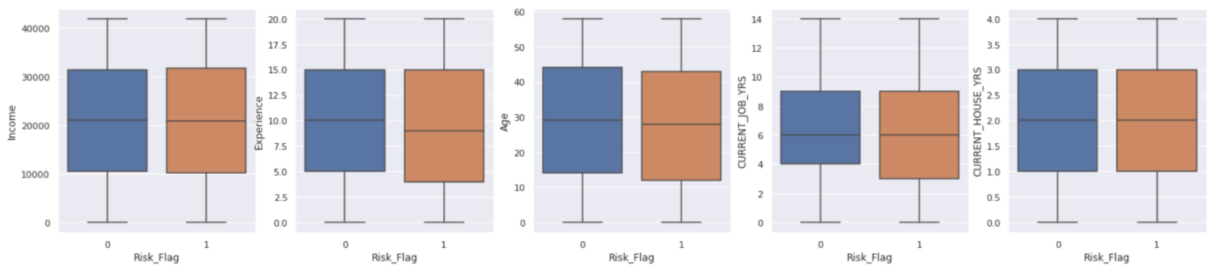


```
plt.figure(figsize=(25,8))
sns.lineplot(df['Age'], df['Experience'], hue=df['Risk_Flag'],ci=0)
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments wi
  warnings.warn(
<matplotlib.axes._subplots.AxesSubplot at 0x7f9ca2e8c100>
```

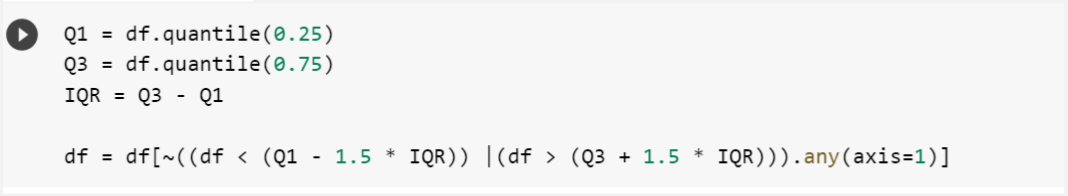


```
f = plt.figure(figsize=(25,5))
ax = f.subplots(1, 5)
sns.boxplot(x='Risk_Flag', y='Income', data=df, ax=ax[0])
sns.boxplot(x='Risk_Flag', y='Experience', data=df, ax=ax[1])
sns.boxplot(x='Risk_Flag', y='Age', data=df, ax=ax[2])
sns.boxplot(x='Risk_Flag', y='CURRENT_JOB_YRS', data=df, ax=ax[3])
sns.boxplot(x='Risk_Flag', y='CURRENT_HOUSE_YRS', data=df, ax=ax[4])
;
```

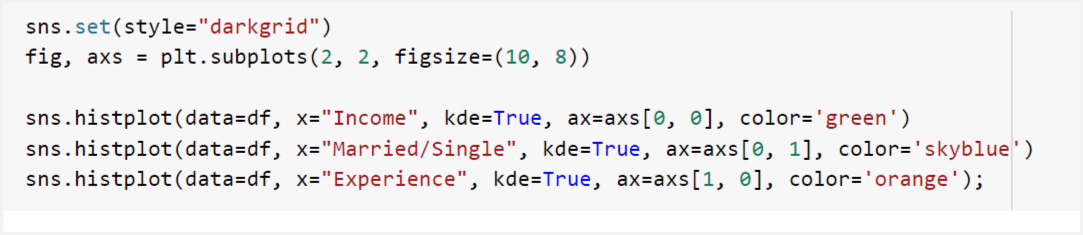


Income, experience, age, current job years, current house years, and non-defaulters can't be distinguished from one other in a substantial way.

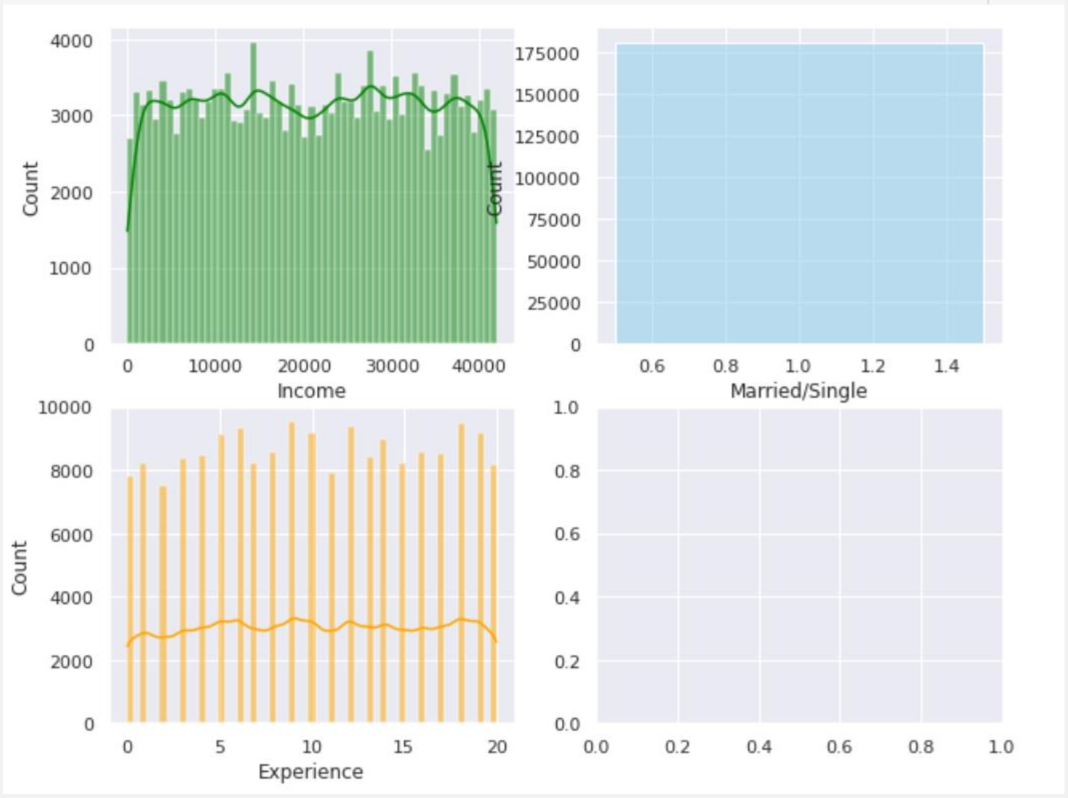
The following stage is to remove outliers and infinite numbers. To do this, we investigate the codes listed below:



```
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1

df = df[~((df < (Q1 - 1.5 * IQR)) |(df > (Q3 + 1.5 * IQR))).any(axis=1)]
```



```
sns.set(style="darkgrid")
fig, axs = plt.subplots(2, 2, figsize=(10, 8))

sns.histplot(data=df, x="Income", kde=True, ax=axs[0, 0], color='green')
sns.histplot(data=df, x="Married/Single", kde=True, ax=axs[0, 1], color='skyblue')
sns.histplot(data=df, x="Experience", kde=True, ax=axs[1, 0], color='orange');
```



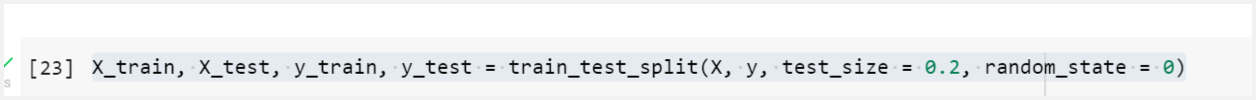
Data Normalization ⚖️

The range of independent variables or data features will be normalised in this part through the process of data normalisation.

Experiments, Evaluation, Analysis and Result

Splitting Data Set

80% of the data set will be used for training and 20% for testing.



```
[23]  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

We can now use our various models to address the challenges we were given to answer after dividing our data into train and test sets.

We will use the several models indicated below for this report:

1.Logistic Regression :

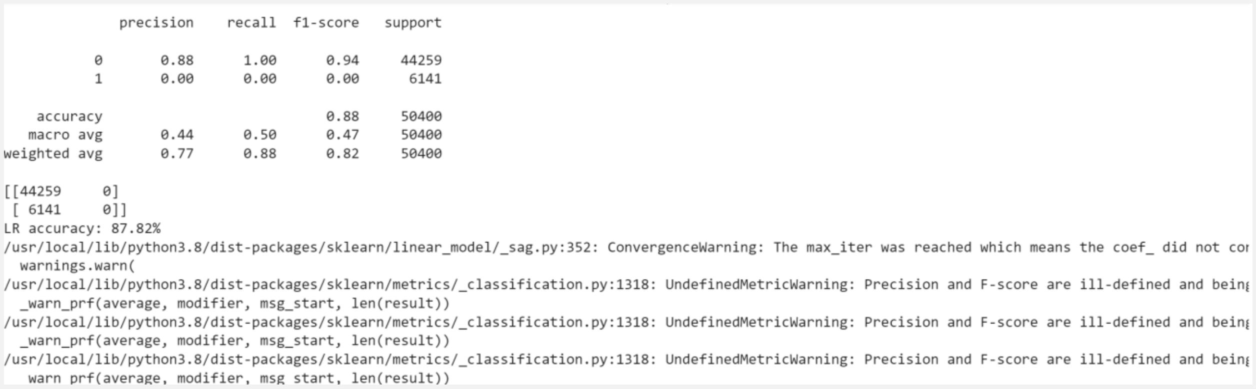
First, we will classify applicants based on their likelihood of defaulting on the loan (Defaulter/non-Defaulter) using the logistic regression model, which is a classification problem.

A type of supervised machine learning called logistic regression is used to estimate or forecast the likelihood that a binary event will occur (yes/no in this case, defaulter/non-defaulter).

Binary, ordinal, and multinomial logistic regression all have different methods of operation and theoretical underpinnings.

The following codes work with this model:

```
LRclassifier = LogisticRegression(solver='saga', max_iter=500, random_state=1)
LRclassifier.fit(X_train, y_train)

y_pred = LRclassifier.predict(X_test)

print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))

from sklearn.metrics import accuracy_score
LRAcc = accuracy_score(y_pred,y_test)
print('LR accuracy: {:.2f}%'.format(LRAcc*100))
```



```
              precision    recall  f1-score   support

           0       0.88      1.00      0.94     44259
           1       0.00      0.00      0.00      6141

    accuracy                           0.88     50400
   macro avg       0.44      0.50      0.47     50400
weighted avg       0.77      0.88      0.82     50400

[[44259     0]
 [ 6141     0]]
LR accuracy: 87.82%
/usr/local/lib/python3.8/dist-packages/sklearn/linear_model/_sag.py:352: ConvergenceWarning: The max_iter was reached which means the coef_ did not co
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/metrics/_classification.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined and bein
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.8/dist-packages/sklearn/metrics/_classification.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined and bein
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.8/dist-packages/sklearn/metrics/_classification.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined and bein
  _warn_prf(average, modifier, msg_start, len(result))
```

As can be seen, our logistic regression model has a respectable accuracy of 87.82%.

2.Random Forest :

The random forests or random decision forests ensemble learning strategy, which is used for classification, regression, and other tasks, builds a lot of decision trees during the training phase. The result of the random forest for classification issues is the class that the majority of the trees choose. For regression tasks, the mean or average forecast of each individual tree is returned.

Random choice forests correct the tendency of decision trees to overfit their training set. Gradient boosted trees are more precise than random forests, often outperforming decision trees. However, data anomalies may reduce their usefulness.

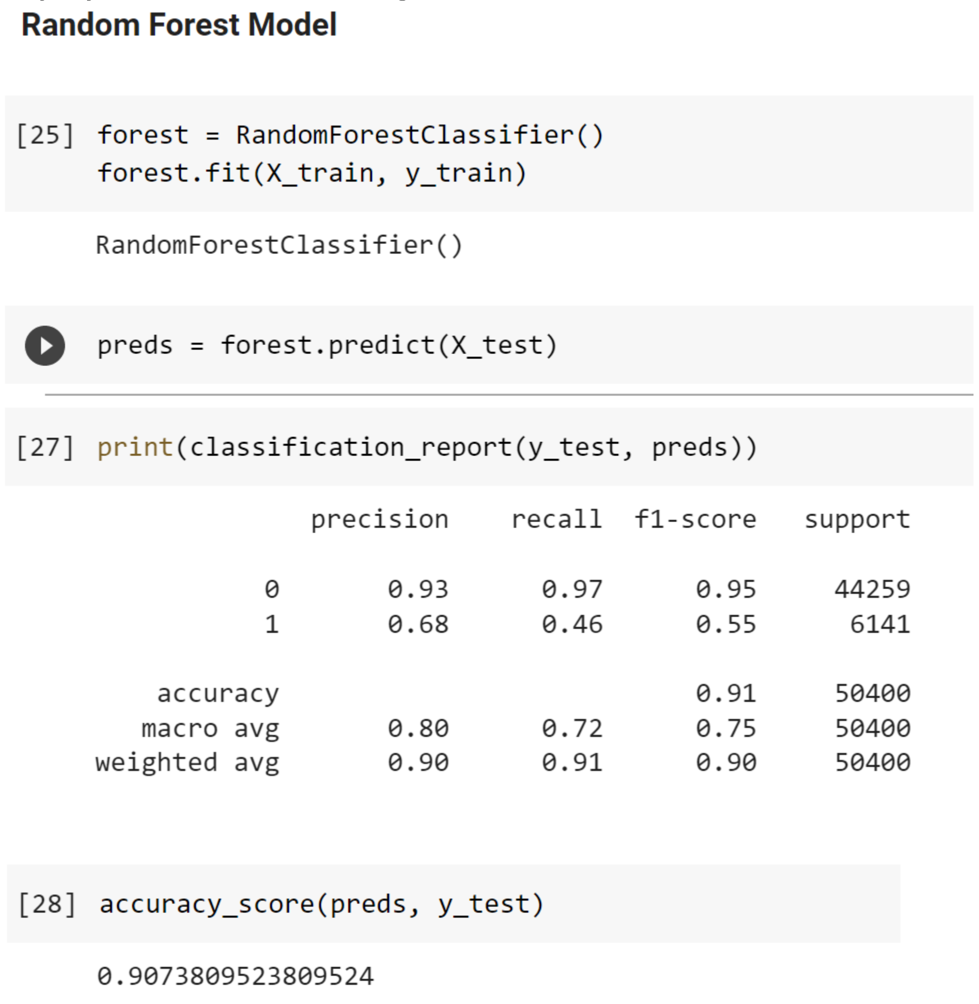
The bootstrap aggregating, or bagging, method is a general strategy that is applied to tree learners by the random forest training algorithm.

Given a training set X = x1, ..., xn with responses Y = y1, ..., yn, bagging repeatedly (B times) selects a random sample with replacement of the training set and fits trees to these samples:

For b = 1, ..., B:

1. Sample, with replacement, n training examples from X, Y; call these Xb, Yb.
2. Train a classification or regression tree fb on Xb, Yb.

Following training, predictions for unobserved samples x' can be made by averaging the predictions from each individual regression tree on x', or in the case of classification trees, by choosing the majority vote. The code following is used to run the random forest classifier:



Random Forest Model

```
[25] forest = RandomForestClassifier()
     forest.fit(X_train, y_train)

     RandomForestClassifier()
```

```
preds = forest.predict(X_test)
```

```
[27] print(classification_report(y_test, preds))

                   precision    recall  f1-score   support

              0       0.93      0.97      0.95     44259
              1       0.68      0.46      0.55      6141

       accuracy                           0.91     50400
      macro avg       0.80      0.72      0.75     50400
   weighted avg       0.90      0.91      0.90     50400
```

```
[28] accuracy_score(preds, y_test)

     0.9073809523809524
```

As we can see from the results above, our random forest classifier achieves an accuracy of 90.7%, which is a very commendable result.

3.K-Nearest Neighbor (KNN) :

K-nearest neighbour is a preferred model to utilise among others since it can compete with the most accurate models and gives incredibly precise predictions. KNN algorithms can be employed for applications requiring great accuracy but not human-readable models.

The accuracy of the projections is influenced by the distance measurement.

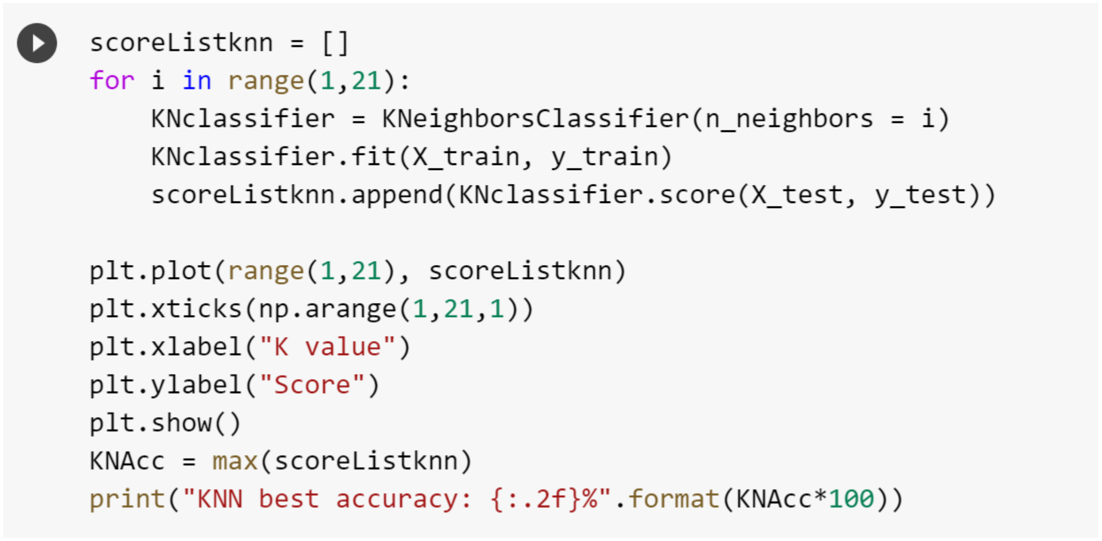
A supervised learning technique called the K-nearest algorithm uses proximity to categorise or anticipate how a single data point will be categorised.

Remember that the KNN approach is a type of "lazy learning" model, which doesn't actually go through a training phase but instead only stores training datasets. Additionally, this suggests that all calculations occur at the same time as making a classification or prediction. It is also known as an instance-based or memory-based learning approach because it significantly relies on memory to retain all of its training data. The separation between the query point and the other data points must be determined in order to determine which data points are closest to a given query point.

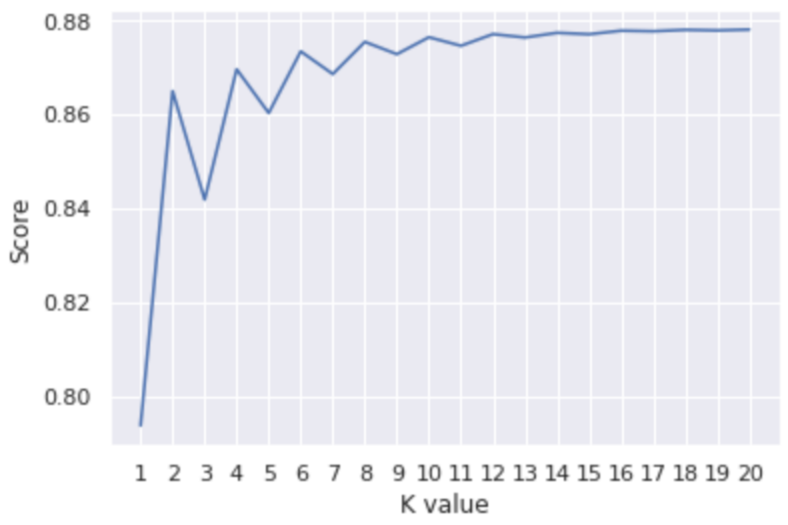
The establishment of decision boundaries, which separate query points into several zones, is made easier by these distance measurements. Voronoi diagrams are widely used to depict decision boundaries.

The K-Nearest Neighbor method is applied in the following code:

K-Nearest Neighbour (KNN)



```
scoreListknn = []
for i in range(1,21):
    KNclassifier = KNeighborsClassifier(n_neighbors = i)
    KNclassifier.fit(X_train, y_train)
    scoreListknn.append(KNclassifier.score(X_test, y_test))

plt.plot(range(1,21), scoreListknn)
plt.xticks(np.arange(1,21,1))
plt.xlabel("K value")
plt.ylabel("Score")
plt.show()
KNAcc = max(scoreListknn)
print("KNN best accuracy: {:.2f}%".format(KNAcc*100))
```



```
KNN best accuracy: 87.81%
```

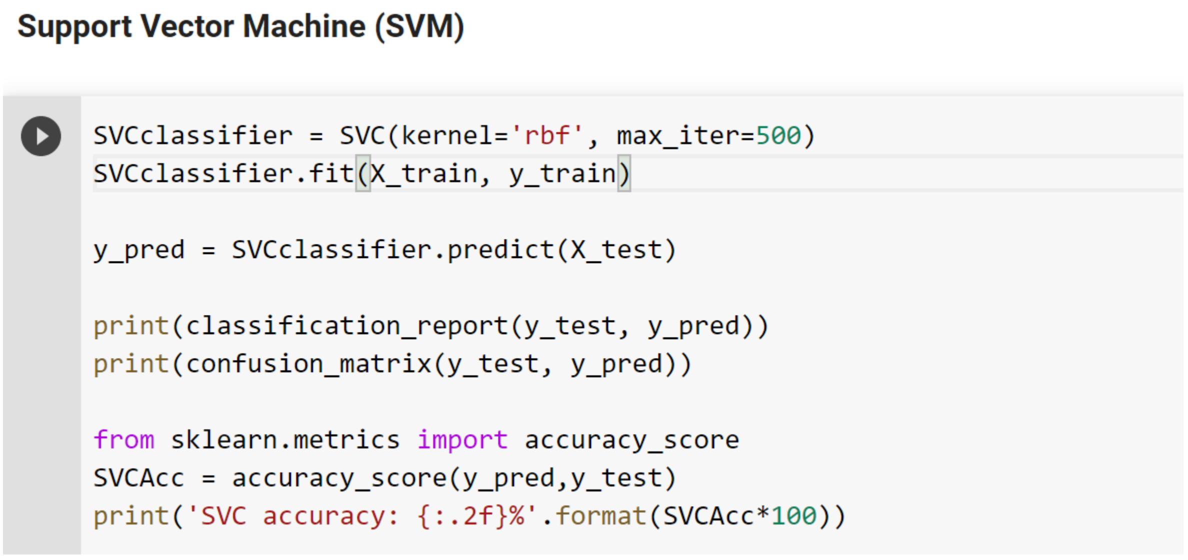
According to the data above, KNN has an accuracy rating of 87.81%, which is likewise a respectable rating.

4.Support Vector Machine (SVM) :

A supervised machine learning approach called "Support Vector Machine" (SVM) can be applied to problems involving classification, regression, and outliers detection. Support vectors are points of data or data point that are closer to the hyperplane and have an effect on the position and orientation of the hyperplane..

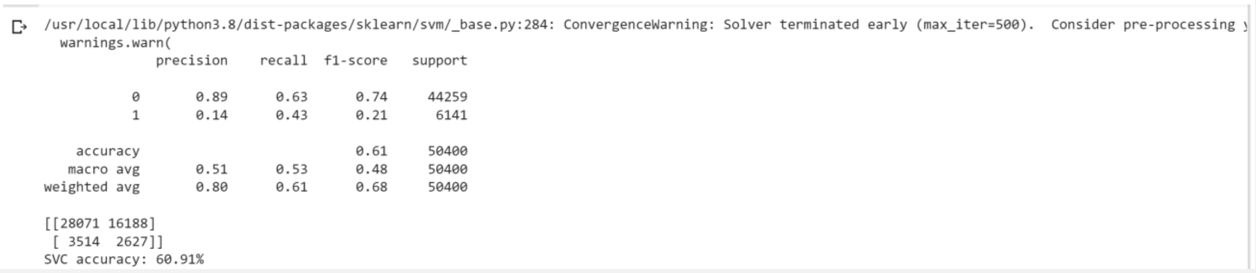
A support vector machine (SVM) is a supervised machine learning method that divides data into two categories and then performs classification or regression tasks using that division. The division that is closest to the limits of each group equally is the best one.

Regression is a less popular application of the technique than categorization. The SVM for regression is also referred to as support vector regression in some publications (SVR). An SVM model is run using the code below:



Support Vector Machine (SVM)

```
SVCclassifier = SVC(kernel='rbf', max_iter=500)
SVCclassifier.fit(X_train, y_train)

y_pred = SVCclassifier.predict(X_test)

print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))

from sklearn.metrics import accuracy_score
SVCAcc = accuracy_score(y_pred,y_test)
print('SVC accuracy: {:.2f}%'.format(SVCAcc*100))
```



```
/usr/local/lib/python3.8/dist-packages/sklearn/svm/_base.py:284: ConvergenceWarning: Solver terminated early (max_iter=500).  Consider pre-processing y
  warnings.warn(
              precision    recall  f1-score   support

           0       0.89      0.63      0.74     44259
           1       0.14      0.43      0.21      6141

    accuracy                           0.61     50400
   macro avg       0.51      0.53      0.48     50400
weighted avg       0.80      0.61      0.68     50400

[[28071 16188]
 [ 3514  2627]]
SVC accuracy: 60.91%
```

We received a very poor accuracy score from our SVM model—60.91%.

5.K-Means Clustering:

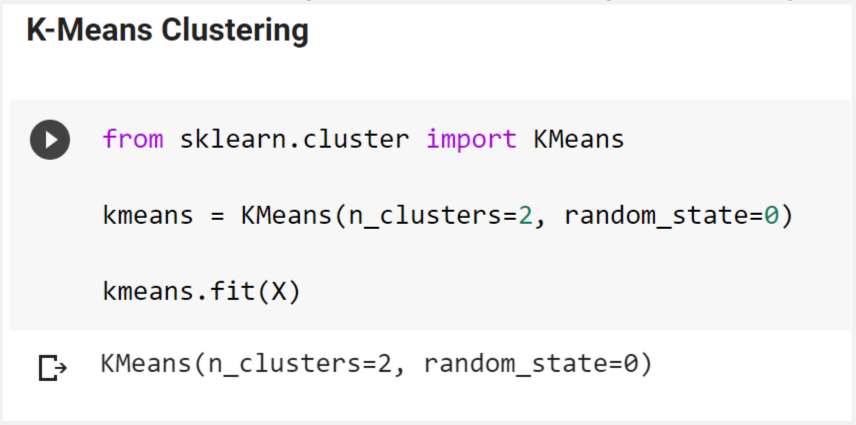
To divide n observations into k clusters, each of which has a prototype (cluster centroid or cluster centre) that each observation belongs to, is the aim of K-means clustering, a vector quantization technique that was first used in signal processing. The result is the division of the data space into Voronoi cells. Euclidean distances can only be minimised by the geometric median; the more difficult Weber problem can only be solved by k-means clustering, which minimises within-cluster variances (squared Euclidean distances). For instance, it is possible to obtain better Euclidean solutions using k-medians and k-medoids.

The k-means technique is used to look for a preset number of clusters in an unlabeled multidimensional dataset. It accomplishes this by using a simple description of what the ideal clustering consists of:

Each point is nearer to its own cluster centre than it is to other cluster centres. The "cluster centre" is the arithmetic mean of all the points that make up the cluster.

These two theories serve as the foundation for the k-means model. Let's look at a simple dataset and the k-means output for the time being before delving into the specific steps the algorithm took to get this conclusion.

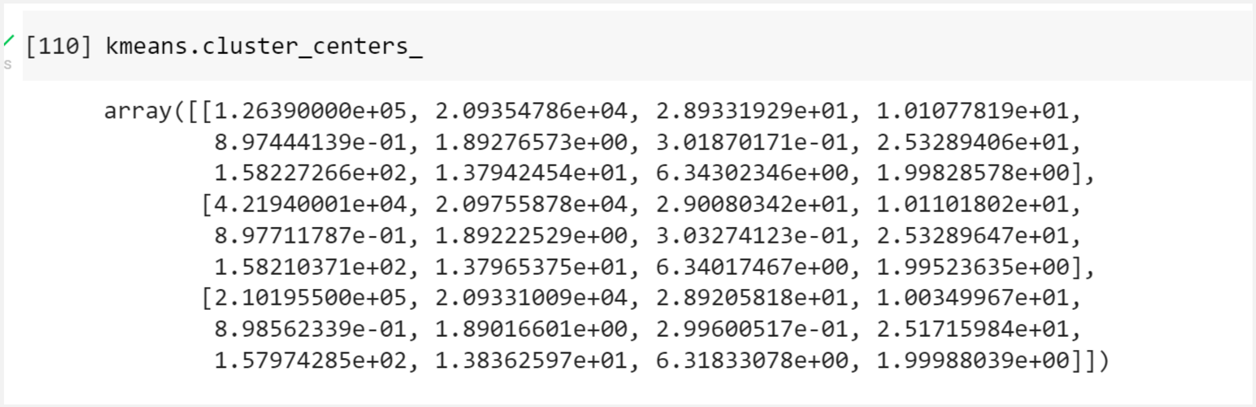
The K-means clustering technique is run using the following code:



K-Means Clustering

```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=2, random_state=0)

kmeans.fit(X)
```

```
KMeans(n_clusters=2, random_state=0)
```



```
[110] kmeans.cluster_centers_
```

```
array([[1.26390000e+05, 2.09354786e+04, 2.89331929e+01, 1.01077819e+01,
        8.97444139e-01, 1.89276573e+00, 3.01870171e-01, 2.53289406e+01,
        1.58227266e+02, 1.37942454e+01, 6.34302346e+00, 1.99828578e+00],
       [4.21940001e+04, 2.09755878e+04, 2.90080342e+01, 1.01101802e+01,
        8.97711787e-01, 1.89222529e+00, 3.03274123e-01, 2.53289647e+01,
        1.58210371e+02, 1.37965375e+01, 6.34017467e+00, 1.99523635e+00],
       [2.10195500e+05, 2.09331009e+04, 2.89205818e+01, 1.00349967e+01,
        8.98562339e-01, 1.89016601e+00, 2.99600517e-01, 2.51715984e+01,
        1.57974285e+02, 1.38362597e+01, 6.31833078e+00, 1.99988039e+00]])
```

• The KMeans algorithm clusters data by attempting to divide samples into n groups with identical variances, while attempting to minimise an indicator referred to as inertia, or within-cluster sum-of-squares. A measure of how internally cohesive clusters are can be found in inertia, also known as the within-cluster sum of squares criterion.

• The k-means technique separates a collection of N samples into K disjoint clusters C, where each cluster's mean j describes its collection of samples.

•

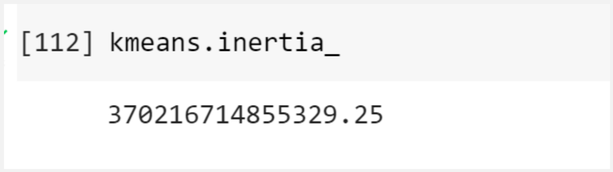
The cluster centroids are another name for the means.

• The within-cluster sum of squared criterion is the goal of the K-means algorithm, which seeks to select centroids that minimise it.

*Inertia: For inertia, there is no standardised metric. Inertia with lower values is preferable, and zero is the best value. But euclidean distances often inflate in very high-dimensional spaces (this is an instance of curse of dimensionality).

This issue can be resolved and the computations sped up by running a dimensionality reduction approach like PCA before k-means clustering.

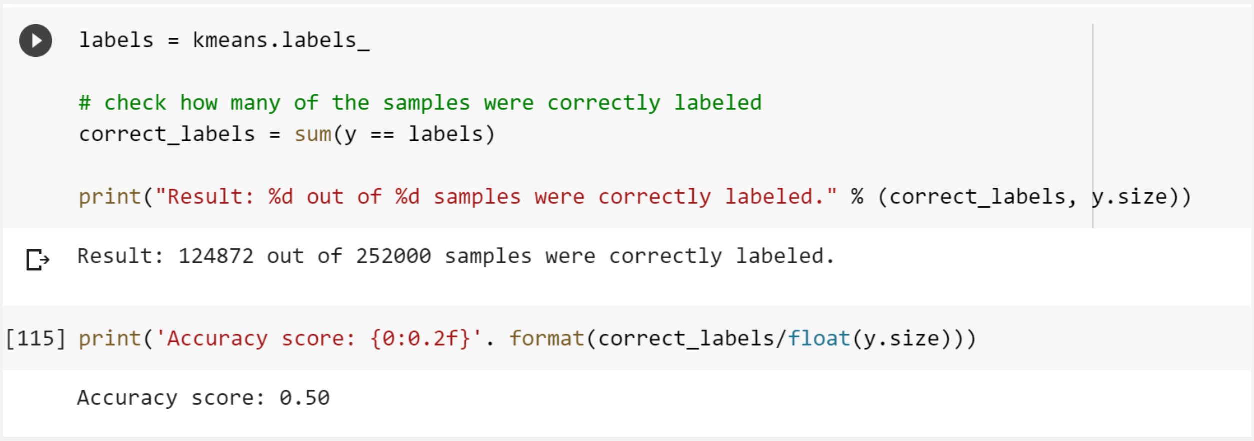
Model inertia can be calculated as follows:



```
[112] kmeans.inertia_

      370216714855329.25
```

The better the model fits, the less inertia there is in the model.

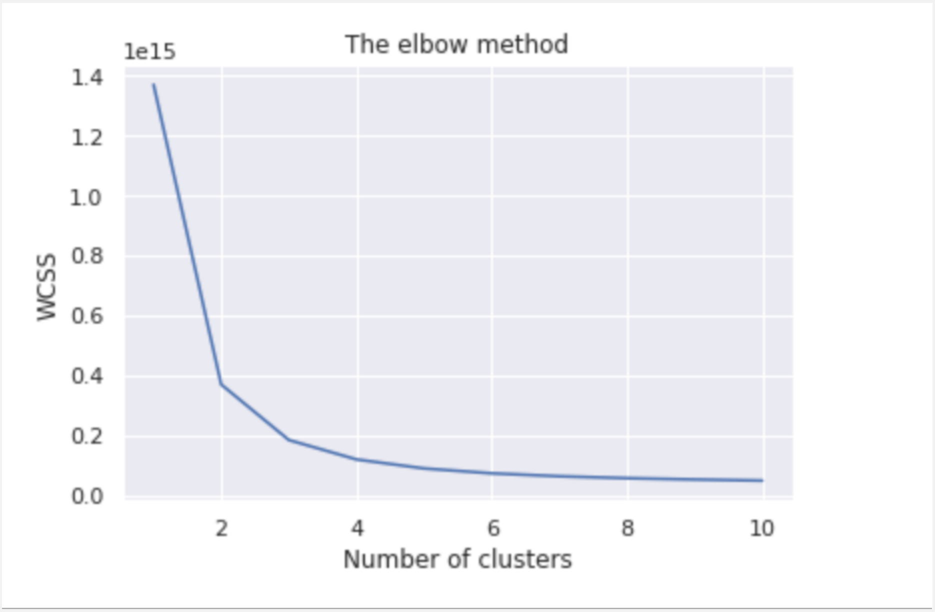
The model has a very high inertia, as is evident. Therefore, this model does not adequately fit the data.



```
labels = kmeans.labels_

# check how many of the samples were correctly labeled
correct_labels = sum(y == labels)

print("Result: %d out of %d samples were correctly labeled." % (correct_labels, y.size))

Result: 124872 out of 252000 samples were correctly labeled.

[115] print('Accuracy score: {0:0.2f}'. format(correct_labels/float(y.size)))

      Accuracy score: 0.50
```

Our unsupervised model has produced a weak classification accuracy of 50%.

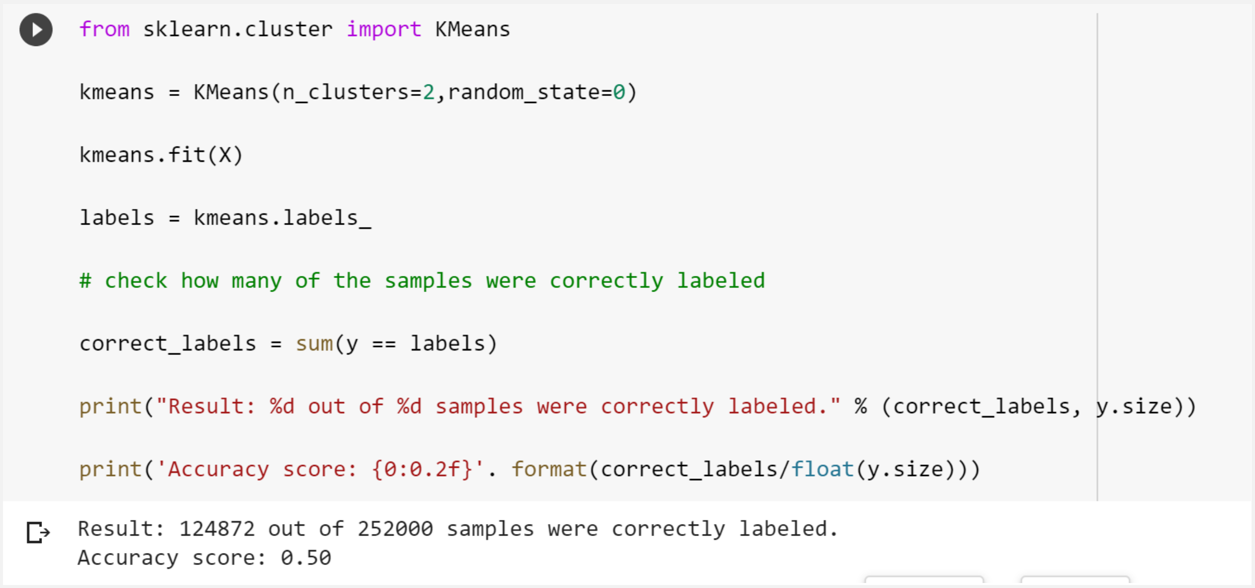
```
#Finding the optimum number of clusters for k-means classification
from sklearn.cluster import KMeans
wcss = []

for i in range(1, 11):

    kmeans = KMeans(n_clusters = i, init = 'k-means++')
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)

#Plotting the results onto a line graph, allowing us to observe 'The elbow'
plt.plot(range(1, 11), wcss)
plt.title('The elbow method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS') #within cluster sum of squares
plt.show()
```



```
[84] kmeans = KMeans(n_clusters = 3, init = 'k-means++')
     y_kmeans = kmeans.fit_predict(X)
```

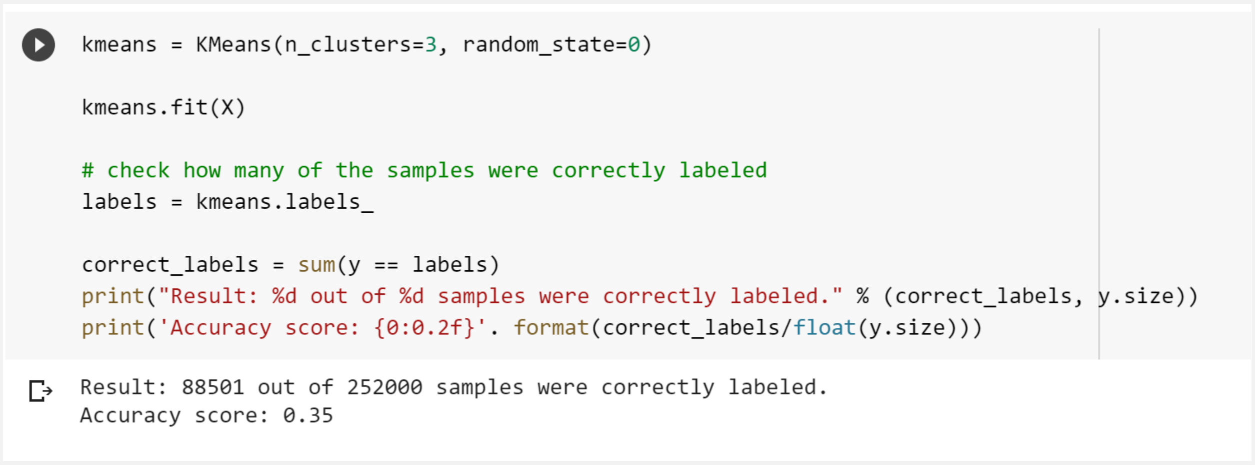
```
print(y_kmeans)
```

```
[1 1 1 ... 2 2 2]
```



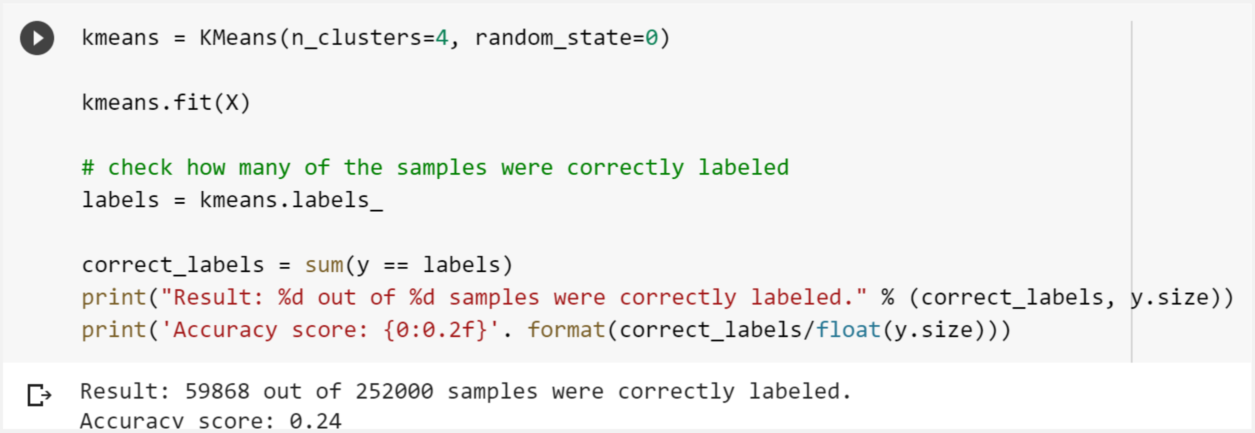
```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=2,random_state=0)

kmeans.fit(X)

labels = kmeans.labels_

# check how many of the samples were correctly labeled

correct_labels = sum(y == labels)

print("Result: %d out of %d samples were correctly labeled." % (correct_labels, y.size))

print('Accuracy score: {0:0.2f}'. format(correct_labels/float(y.size)))
```

```
Result: 124872 out of 252000 samples were correctly labeled.
Accuracy score: 0.50
```



```
kmeans = KMeans(n_clusters=3, random_state=0)

kmeans.fit(X)

# check how many of the samples were correctly labeled
labels = kmeans.labels_

correct_labels = sum(y == labels)
print("Result: %d out of %d samples were correctly labeled." % (correct_labels, y.size))
print('Accuracy score: {0:0.2f}'. format(correct_labels/float(y.size)))
```

```
Result: 88501 out of 252000 samples were correctly labeled.
Accuracy score: 0.35
```



```
kmeans = KMeans(n_clusters=4, random_state=0)

kmeans.fit(X)

# check how many of the samples were correctly labeled
labels = kmeans.labels_

correct_labels = sum(y == labels)
print("Result: %d out of %d samples were correctly labeled." % (correct_labels, y.size))
print('Accuracy score: {0:0.2f}'. format(correct_labels/float(y.size)))
```

```
Result: 59868 out of 252000 samples were correctly labeled.
Accuracy score: 0.24
```

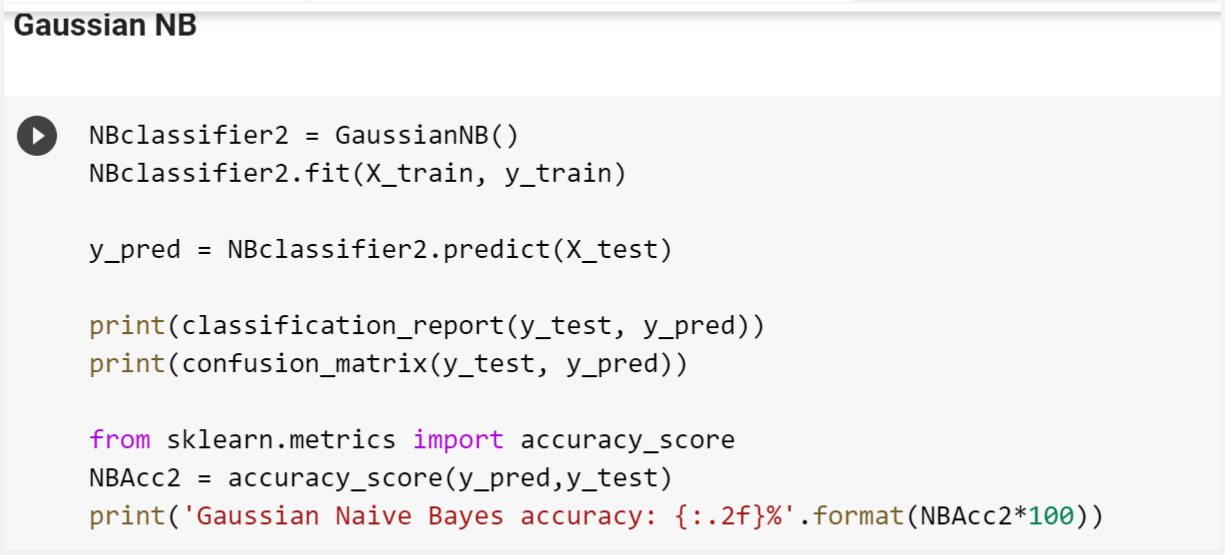
6.Gaussian Naïve Bayes :

The application of the Bayes theorem with stringent independence conditions is the foundation of the probabilistic classification technique known as Gaussian Naive Bayes. When categorising anything, the term "independence" refers to the idea that the existence of one value for a feature has no influence on the existence of another (unlike independence in probability theory). It's considered naïve to believe that an object's features are unconnected to one another. In the domain of machine learning, naive Bayes classifiers are well known to be highly expressive, scalable, and fairly accurate, but their performance quickly deteriorates with the increase of the training set. A variety of factors can affect how well naive Bayes classifiers perform.

The parameters of the classification model don't need to be changed, and they can easily handle continuous features and scale well with the size of the training data set. These are their primary benefits.

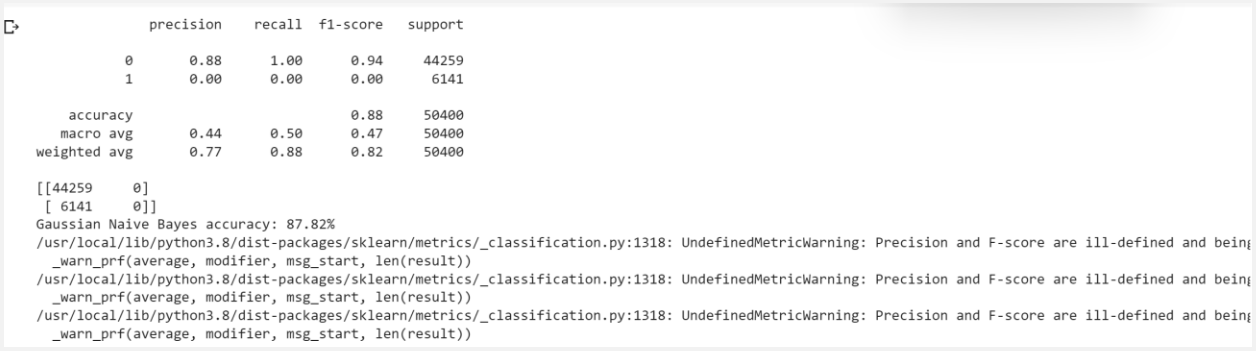
Gaussian Naive Bayes is a variant of Naive Bayes that adheres to the Gaussian normal distribution and supports continuous data.

The Gaussian Naive Bayes algorithm is run using the code below:



Gaussian NB

```
NBclassifier2 = GaussianNB()
NBclassifier2.fit(X_train, y_train)

y_pred = NBclassifier2.predict(X_test)

print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))

from sklearn.metrics import accuracy_score
NBAcc2 = accuracy_score(y_pred,y_test)
print('Gaussian Naive Bayes accuracy: {:.2f}%'.format(NBAcc2*100))
```



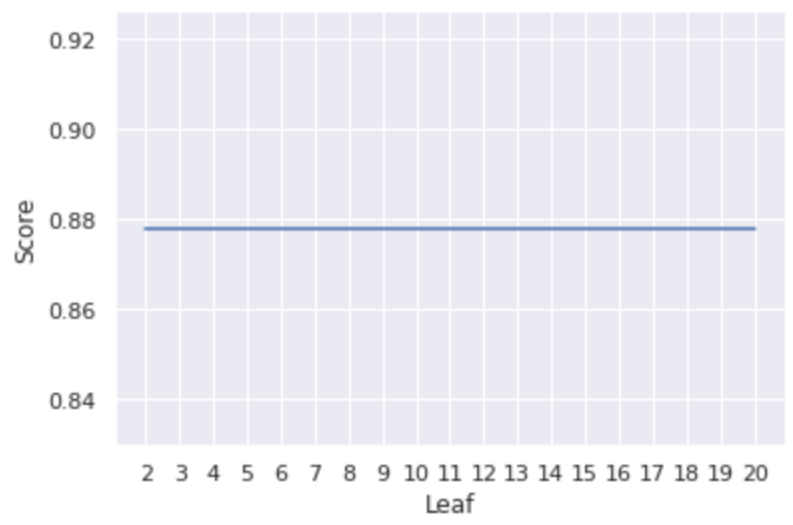
```
              precision    recall  f1-score   support

           0       0.88      1.00      0.94     44259
           1       0.00      0.00      0.00      6141

    accuracy                           0.88     50400
   macro avg       0.44      0.50      0.47     50400
weighted avg       0.77      0.88      0.82     50400

[[44259     0]
 [ 6141     0]]
Gaussian Naive Bayes accuracy: 87.82%
/usr/local/lib/python3.8/dist-packages/sklearn/metrics/_classification.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined and bein
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.8/dist-packages/sklearn/metrics/_classification.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined and bein
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.8/dist-packages/sklearn/metrics/_classification.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined and bein
  _warn_prf(average, modifier, msg_start, len(result))
```

7.Decision Trees :

The non-parametric supervised learning approach used for classification and regression applications is the decision tree. It is organised hierarchically and has a root node, branches, internal nodes, and leaf nodes.

Decision Tree

```
scoreListDT = []
for i in range(2,21):
    DTclassifier = DecisionTreeClassifier(max_leaf_nodes=i)
    DTclassifier.fit(X_train, y_train)
    scoreListDT.append(DTclassifier.score(X_test, y_test))

plt.plot(range(2,21), scoreListDT)
plt.xticks(np.arange(2,21,1))
plt.xlabel("Leaf")
plt.ylabel("Score")
plt.show()
DTAcc = max(scoreListDT)
print("Decision Tree Accuracy: {:.2f}%".format(DTAcc*100))
```



```
Decision Tree Accuracy: 87.82%
```

Our decision tree method has a good accuracy of 87.82%.

8.XGBoost :

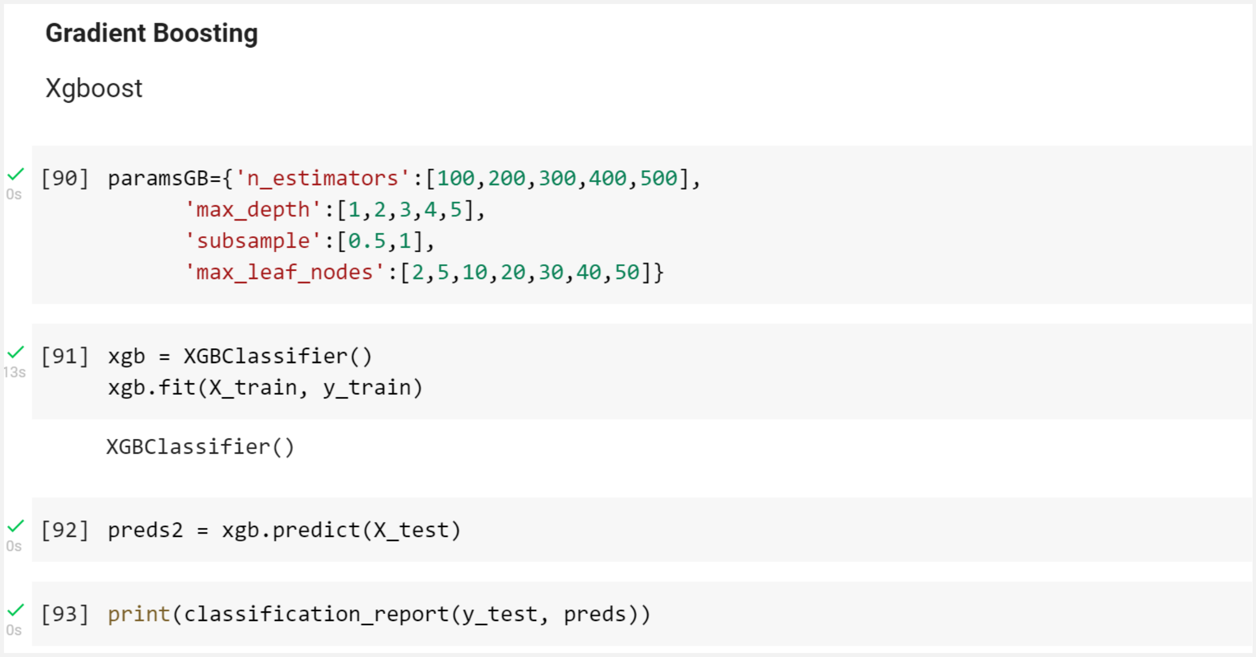
A distributed, scalable gradient-boosted decision tree (GBDT) machine learning framework is called Extreme Gradient Boosting (XGBoost). It provides parallel tree boosting and is the best machine learning package for regression, classification, and ranking problems.

Understanding supervised machine learning, decision trees, ensemble learning, and gradient boosting, as well as XGBoost, requires a thorough understanding of the principles and methods of machine learning.

In supervised machine learning, a model is trained using algorithms to find trends in a dataset of features and labels, after which the model is applied to forecast the labels on the features of a new dataset.

Gradient Boosting Decision Trees (GBDT), a decision tree ensemble learning method comparable to random forest, are used for classification and regression. Ensemble learning algorithms combine many machine learning approaches to produce more accurate models.

Both GBDT and random forest create a model made up of several decision trees. The way the trees are constructed and joined makes a difference.



Gradient Boosting

Xgboost

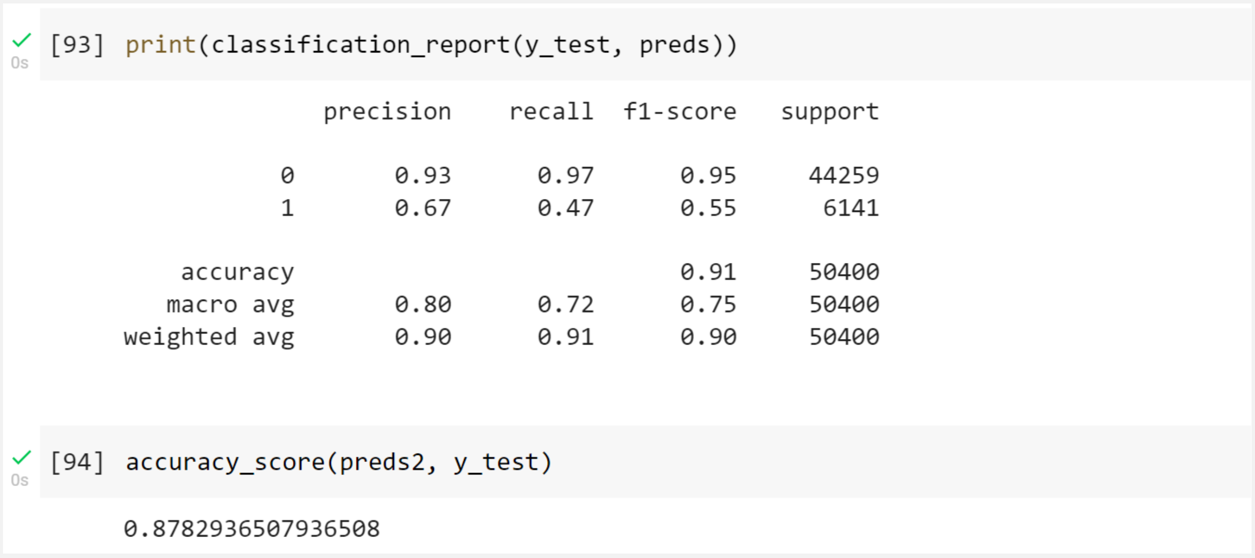
```
[90] paramsGB={'n_estimators':[100,200,300,400,500],
            'max_depth':[1,2,3,4,5],
            'subsample':[0.5,1],
            'max_leaf_nodes':[2,5,10,20,30,40,50]}
```

```
[91] xgb = XGBClassifier()
     xgb.fit(X_train, y_train)
```

```
XGBClassifier()
```

```
[92] preds2 = xgb.predict(X_test)
```

```
[93] print(classification_report(y_test, preds))
```



```
[93] print(classification_report(y_test, preds))

              precision    recall  f1-score   support

           0       0.93      0.97      0.95     44259
           1       0.67      0.47      0.55      6141

    accuracy                           0.91     50400
   macro avg       0.80      0.72      0.75     50400
weighted avg       0.90      0.91      0.90     50400
```

```
[94] accuracy_score(preds2, y_test)

0.8782936507936508
```

Conclusions

Eight distinct models were used to train our dataset across three different problems: classification, clustering, and regression. 1. Decision Trees, 2. Logistic Regression, 3. Gaussian NB, 4. K-Nearest Neighbor, 5. K-Means clustering, 6. SVM, 7.XGBoost, 8.Random Forest,

These models allowed us to attain the following accuracy levels:

1.Logistic Regression - 87.82%
2.Random Forest - 90.73%
3.K-Nearest Neighbor - 87.81%
4.Support Vector Mechanism - 60.91%
5.K-Means – 50%
6.Gaussian Naïve Bayes - 87.82%
7.Decision Trees - 87.82%
8.XGBoost - 87.82%

According to the accuracy report above, random forest provided the best prediction accuracy, while K-means provided the worst.

Real-world classification issues are typically unbalanced. Additionally, data sets typically contain some missing values. We discussed methods for handling both missing values and unbalanced data sets in this post. Additionally, we looked at various sklearn ensemble construction strategies. Here are some key ideas to remember:

There is no set rule for which algorithms to apply in what circumstances. What may be effective on some data sets may not always be effective on others. To obtain accurate estimations, constantly evaluate your procedures using cross validation.

In some cases, we could be willing to give up some model improvement if doing so would significantly increase complexity relative to the improvement in assessment measures.

False Negatives can be significantly more expensive than False Positives in certain categorization issues. Consequently, we can lower cut-off points to lower False Negatives.

Use high-quality models that are as diverse from one another as you can while creating ensemble models to lessen correlation between the base learners. We may have improved our stacked ensemble model by including Dense Neural Network and other types of base learners in addition to more layers.

In general, EasyEnsemble outperforms all other resampling techniques.

Sometimes missing values provide the model more information than we might anticipate. In order to determine whether an example is missing or not, it is possible to capture it by adding binary features to each feature that has missing values.

References

SURANA. (n.d.). *Loan Prediction Based on Customer Behavior*. Retrieved December 20, 2022, from https://www.kaggle.com/datasets/subhamjain/loan-prediction-based-on-customer-behavior/code?resource=download

FEILIU. (2022, August). *MM1041 Supervised Machine Learning*. MM1041 Supervised Machine Learning | Kaggle. Retrieved December 22, 2022, from https://www.kaggle.com/code/holybread/mm1041-supervised-machine-learning

Raschka, Liu, Mijalili, & et al. (2022, February 25). Machine Learning with PyTorch and Scikit-Learn. In *Machine Learning with PyTorch and Scikit-Learn: Develop machine learning and deep learning models with Python*. Sebastian Raschka , Yuxi (Hayden) Liu , et al.

Huyen. (2022, May 31). Designing Machine Learning Systems: An Iterative Process for Production-Ready Applications. In *Designing Machine Learning Systems: An Iterative Process for Production-Ready Applications*. Chip Huyen.

Neagoie, Bourke, & Mastery. (n.d.). *Complete Machine Learning & Data Science Bootcamp 2023*. udemy.com. Retrieved January 12, 2023, from https://www.udemy.com/course/complete-machine-learning-and-data-science-zero-to-mastery/

Fred Nwangaga. (2022, May 23). *Machine Learning with Python: k-Means Clustering*. LinkedIn Learning. Retrieved January 12, 2023, from https://www.linkedin.com/learning/machine-learning-with-python-k-means-clustering/how-to-segment-data-with-k-means-clustering-in-python?autoplay=true&resume=false&u=42288921

Edureka. (2020, March 15). *Machine Learning Full Course*. https://www.youtube.com/watch?v=GwIo3gDZCVQ. Retrieved January 12, 2023, from https://Machine Learning Full Course - Learn Machine Learning 10 Hours | Machine Learning Tutorial | Edureka