Support Vector Machines - Bank Customers Retirement Predictions Project

Nawal Obaid

9/7/2022

PROBLEM STATEMENT

You work as a data scientist at a major bank in NYC and you have been tasked to develop a model that can predict whether a customer is able to retire or not based on his/her features. Features are his/her age and net 401K savings (retirement savings in the U.S.). You thought that Support Vector Machines can be a great candidate to solve the problem.

Import Needed Libraries

In [97]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

Load Data

In [98]:
```
bank_df = pd.read_csv('Bank_Customer_retirement.csv')
```

In [99]:
```
bank_df.head()
# need to do scaling to this dataset
# need to use SVM as a classifier
```

Out[99]:

	Customer ID	Age	401K Savings	Retire
0	0	39.180417	322349.8740	0
1	1	56.101686	768671.5740	1
2	2	57.023043	821505.4718	1
3	3	43.711358	494187.4850	0
4	4	54.728823	691435.7723	1

In [100…
```
bank_df.info()
#500 entries,(total 4 columns)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Customer ID   500 non-null    int64  
 1   Age           500 non-null    float64
 2   401K Savings  500 non-null    float64
 3   Retire        500 non-null    int64  
```

```
dtypes: float64(2), int64(2)
memory usage: 15.8 KB
```

In [101…
```
bank_df.shape
```

Out[101]:
```
(500, 4)
```

In [102…
```
bank_df.describe()
```

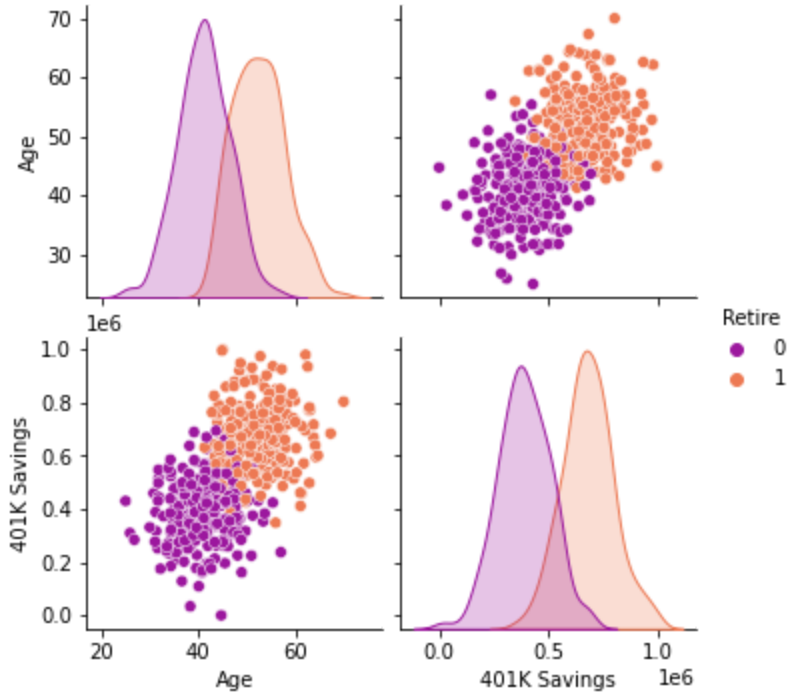
Out[102]:

	Customer ID	Age	401K Savings	Retire
count	500.000000	500.000000	500.000000	500.000000
mean	249.500000	46.757077	534984.465804	0.500000
std	144.481833	7.790125	187675.818881	0.500501
min	0.000000	25.000000	10.000000	0.000000
25%	124.750000	41.299451	382626.524425	0.000000
50%	249.500000	46.695770	534512.984350	0.500000
75%	374.250000	52.322551	680670.257025	1.000000
max	499.000000	70.000000	1000000.000000	1.000000

EDA

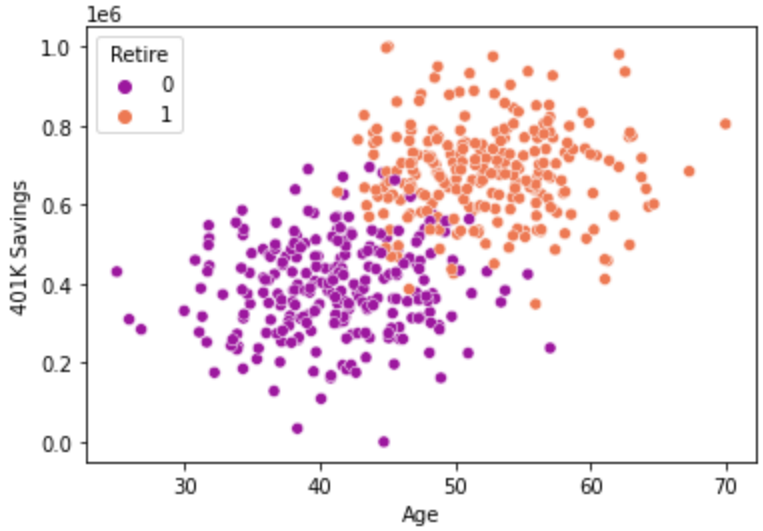
In [103…
```
sns.pairplot(bank_df, hue = 'Retire', vars = ['Age', '401K Savings'], palette = 'plasma'
```

Out[103]:
```
<seaborn.axisgrid.PairGrid at 0x272d20b7910>
```



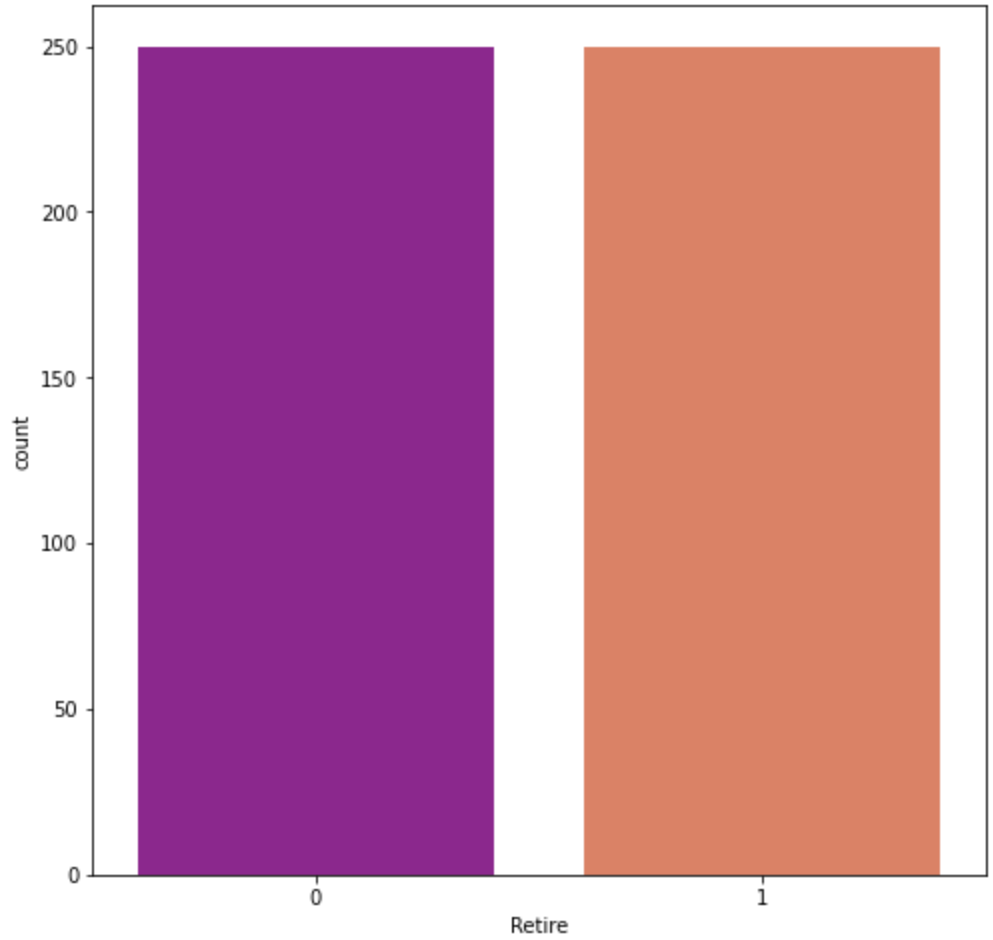
In [104…
```
sns.scatterplot(data= bank_df, x='Age', y='401K Savings', hue = 'Retire', palette = 'pla
```

Out[104]:
```
<AxesSubplot:xlabel='Age', ylabel='401K Savings'>
```



In [105…
```
plt.figure(figsize=[8, 8])
sns.countplot( data = bank_df, x='Retire', palette='plasma')
```

Out[105]:
```
<AxesSubplot:xlabel='Retire', ylabel='count'>
```



In [106…
```
# Let's explore which dataset is missing
sns.heatmap(bank_df.isnull(), yticklabels = False, cbar = False, cmap="Blues")
# No missing values
# No data imputation needed
```

Out[106]:
```
<AxesSubplot:>
```

In [107…
```
# Let's drop the Id coloumn
bank_df.drop('Customer ID',axis=1,inplace=True)
```

In [108…
```
bank_df.head(2)
```

Out[108]:

	Age	401K Savings	Retire
0	39.180417	322349.874	0
1	56.101686	768671.574	1

In [109…
```
X = bank_df.drop('Retire', axis =1)
y = bank_df['Retire']
```

Feature Scaling/Normalization

In [110…
```
# Scaling is needed
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X = scaler.fit_transform(X)
```

Train Test Split

In [111…
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=5)
```

Teain/Fit the model

In [112…
```
from sklearn.svm import SVC
```

In [113…
```
classifier = SVC()
```

In [114…
```
classifier.fit(X_train, y_train)
```

Out[114]:

▾ SVC

SVC()

In [115…
```
predictions = classifier.predict(X_test)
```

In [116…
```
from sklearn.metrics import classification_report, confusion_matrix
```

In [117…
```
print('**************Classification Report*************************')
print(classification_report(y_test, predictions))
print('***************Confusion Matrix*****************************')
print(confusion_matrix(y_test, predictions))
```

```
**************Classification Report*************************
              precision    recall  f1-score   support

           0       0.95      0.93      0.94        43
           1       0.95      0.96      0.96        57

    accuracy                           0.95       100
   macro avg       0.95      0.95      0.95       100
weighted avg       0.95      0.95      0.95       100

***************Confusion Matrix*****************************
[[40  3]
 [ 2 55]]
```

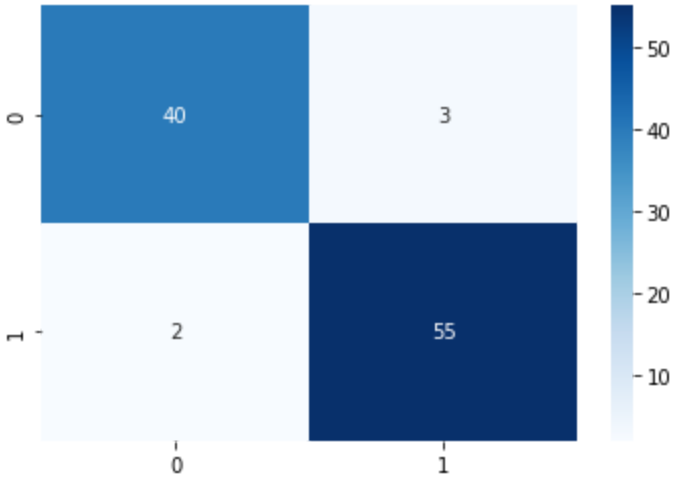
The confusion matrix shows:

False Positives(Type I Error)= 3

False Negatives(Type II Error)= 2

In [119…
```
cm = confusion_matrix(y_test, predictions)
sns.heatmap(cm, annot = True, fmt = "d", cmap ='Blues')
```

Out[119]:
```
<AxesSubplot:>
```



In [ ]: