

REPORT

IML Lab 6 - Task 2

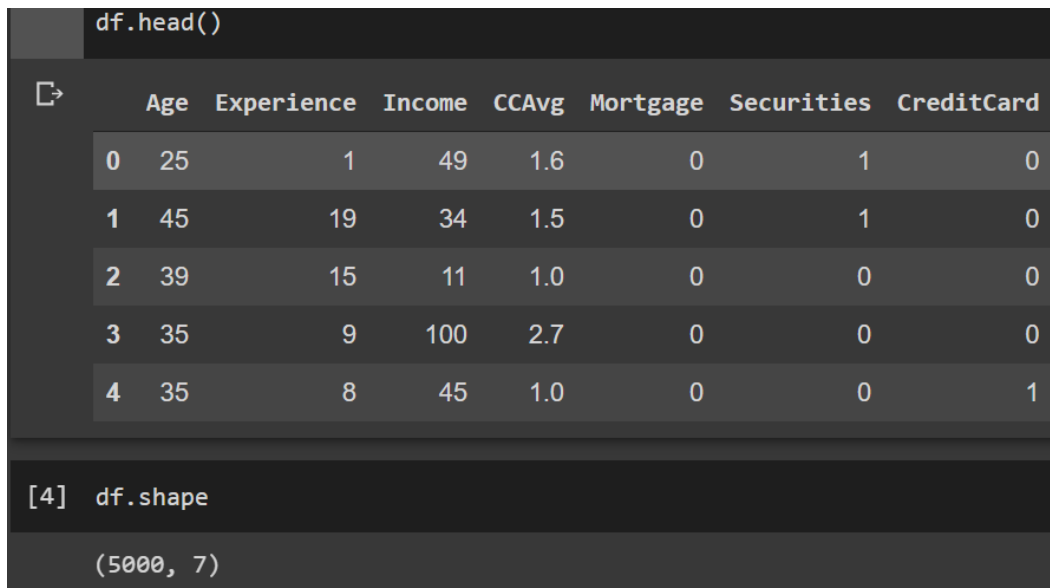
Vrushali Pandit B20BB047

Data :

Based on features of customers such as Age, Income, Experience, CCAvg, Mortgage and Securities we have to predict whether they will buy a credit card or not.

Steps:

1. Importing relevant libraries
2. Mounting google drive, reading the data using Pandas, printing the head and shape of the dataset



```
df.head()
```

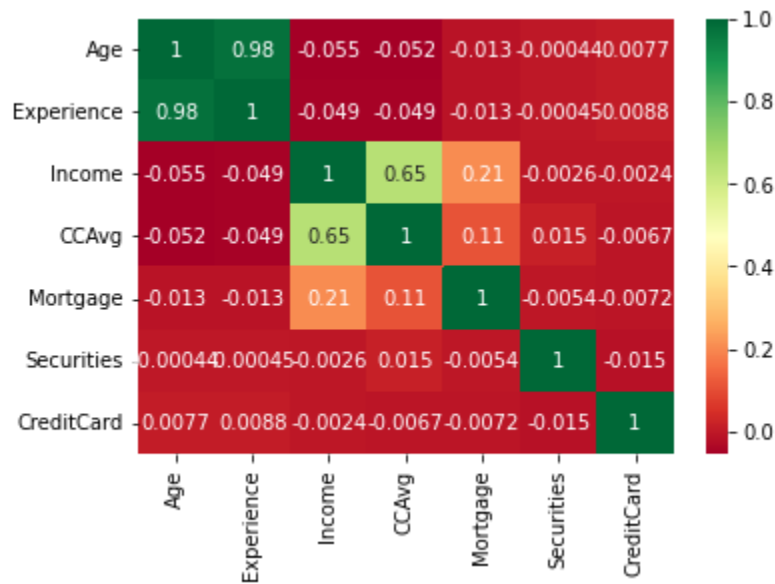
	Age	Experience	Income	CAvg	Mortgage	Securities	CreditCard
0	25	1	49	1.6	0	1	0
1	45	19	34	1.5	0	1	0
2	39	15	11	1.0	0	0	0
3	35	9	100	2.7	0	0	0
4	35	8	45	1.0	0	0	1

```
[4] df.shape
```

```
(5000, 7)
```

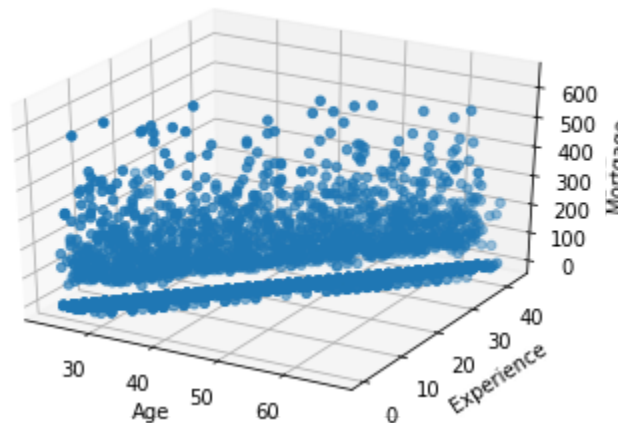
3. Some of the entries were negative integers which is not logical for features like "Experience"
 - a. Loop through the entire dataframe
 - b. Impute these negative noise by the mean
4. We must select any three features to train our model on. The method I chose for feature selection - Highest Correlation amongst the features and the target.

a. Print the correlation matrix for the same



b. We see that "CreditCard" (target variable) has the highest correlation (whether direct or indirect) with "Age", "Experience" and "Mortgage".

c. Make 3D plot of these features



d. Store these three features in a feature-vector, and target into a target vector.

e. Standardize the feature vector using StandardScaler() from sklearn.preprocessing

f. Split the data into 80:20

g. Loop through values of C = [0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000]

i. For each, define the model as

- ii. Fit it
- iii. Print the score on testing data
- iv. Predict the classes for testing data
- v. And print the confusion matrices.

```
from sklearn.svm import LinearSVC
from sklearn.metrics import classification_report, confusion_matrix

for c in [0.0001, 0.001, 0.01, 0.1, 1, 10,100,1000]:
    print('C = ',c)
    svcclassifier = LinearSVC(C=c , random_state=0, max_iter = 3000)
    svcclassifier.fit(X_train_scaled, y_train)
    print("Score on test data = ", svcclassifier.score(X_test_scaled,y_test))
    y_pred = svcclassifier.predict(X_test_scaled)
    print(confusion_matrix(y_test,y_pred))
    print(classification_report(y_test,y_pred))
```

h. Result obtained (added as screenshots)

C = 0.0001

Score on test data = 0.7008

Confusion Matrix :

[[876 0]

[374 0]]

Report:

	precision	recall	f1-score	support
0	0.70	1.00	0.82	876
1	0.00	0.00	0.00	374
accuracy			0.70	1250
macro avg	0.35	0.50	0.41	1250
weighted avg	0.49	0.70	0.58	1250

C = 0.001

Score on test data = 0.7008

[[876 0]

[374 0]]

	precision	recall	f1-score	support
0	0.70	1.00	0.82	876
1	0.00	0.00	0.00	374
accuracy			0.70	1250
macro avg	0.35	0.50	0.41	1250
weighted avg	0.49	0.70	0.58	1250

C = 0.01

Score on test data = 0.7008

[[876 0]

[374 0]]

	precision	recall	f1-score	support
0	0.70	1.00	0.82	876
1	0.00	0.00	0.00	374
accuracy			0.70	1250
macro avg	0.35	0.50	0.41	1250
weighted avg	0.49	0.70	0.58	1250

C = 0.1

Score on test data = 0.7008

[[876 0]

[374 0]]

	precision	recall	f1-score	support
0	0.70	1.00	0.82	876
1	0.00	0.00	0.00	374
accuracy			0.70	1250
macro avg	0.35	0.50	0.41	1250
weighted avg	0.49	0.70	0.58	1250

C = 1

Score on test data = 0.7008

[[876 0]

[374 0]]

	precision	recall	f1-score	support
0	0.70	1.00	0.82	876
1	0.00	0.00	0.00	374
accuracy			0.70	1250
macro avg	0.35	0.50	0.41	1250
weighted avg	0.49	0.70	0.58	1250

C = 10

Score on test data = 0.7008

[[876 0]

[374 0]]

	precision	recall	f1-score	support
0	0.70	1.00	0.82	876
1	0.00	0.00	0.00	374
accuracy			0.70	1250
macro avg	0.35	0.50	0.41	1250
weighted avg	0.49	0.70	0.58	1250

C = 100

Score on test data = 0.7008

[[876 0]

[374 0]]

	precision	recall	f1-score	support
0	0.70	1.00	0.82	876
1	0.00	0.00	0.00	374
accuracy			0.70	1250
macro avg	0.35	0.50	0.41	1250
weighted avg	0.49	0.70	0.58	1250

C = 1000

Score on test data = 0.6664

[[806 70]

[347 27]]

	precision	recall	f1-score	support
0	0.70	0.92	0.79	876
1	0.28	0.07	0.11	374
accuracy			0.67	1250
macro avg	0.49	0.50	0.45	1250
weighted avg	0.57	0.67	0.59	1250

Inference of Results:

1)

We find that for different values of C we are getting the same results. This means that no matter what (within a range because $C = 1000$, performs differently and poorly) the width of the margin doesn't affect the hyperplane. This means that there aren't many sparse datapoints. The classes are heavily clustered and perfectly lie in their place with little variation.

2) Another reason could be that the data itself is not linearly separable.

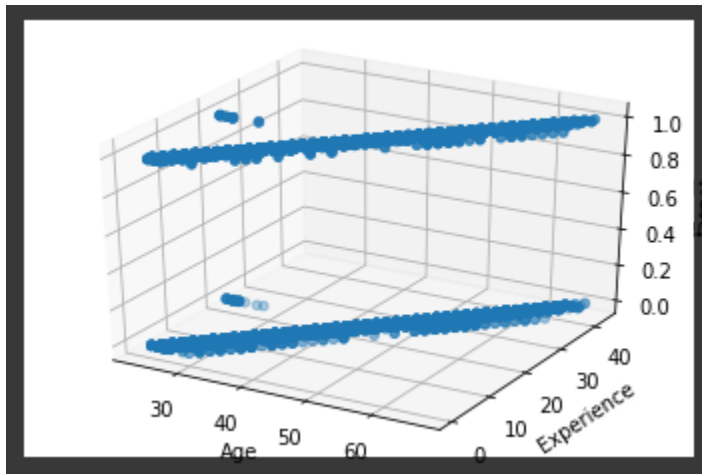
One thing supporting this explanation is that despite taking maximum iterations of SVM as 100,000 (where default = 1000), for some values of C , the model always failed to converge.

However on plotting 3D plots of target with the 2 best correlating features, we see a clear linear separation.

X-axis = age

Y-axis = experience

Z-axis = CreditCards (target variable)



So explanation #1 - that the data is extremely clustered and very few variations is the best explanation for such results.

Colab link =

<https://colab.research.google.com/drive/1k6bZbP8nbRpUrrYu2eoxm4qY1WIA9j3u?usp=sharing>

