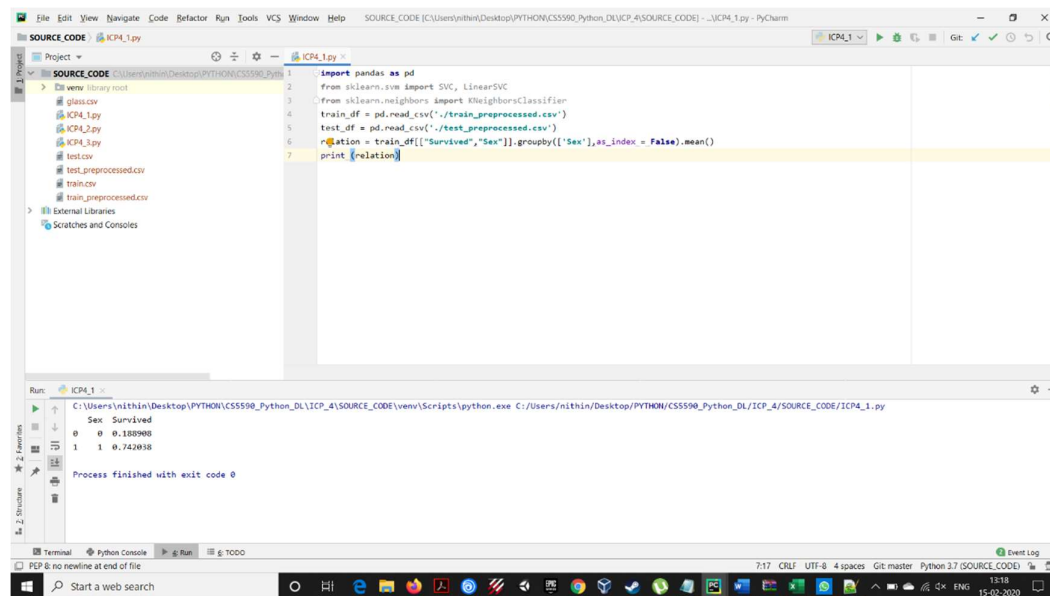


ICP4

1. Find the correlation between 'survived'(target column) and 'sex' column for the Titanic use case in class. Do you think we should keep this feature?



```
1 import pandas as pd
2 from sklearn.svm import SVC, LinearSVC
3 from sklearn.neighbors import KNeighborsClassifier
4 train_df = pd.read_csv("../train_preprocessed.csv")
5 test_df = pd.read_csv("../test_preprocessed.csv")
6 relation = train_df[["Survived", "Sex"]].groupby(["Sex"], as_index = False).mean()
7 print (relation)
```

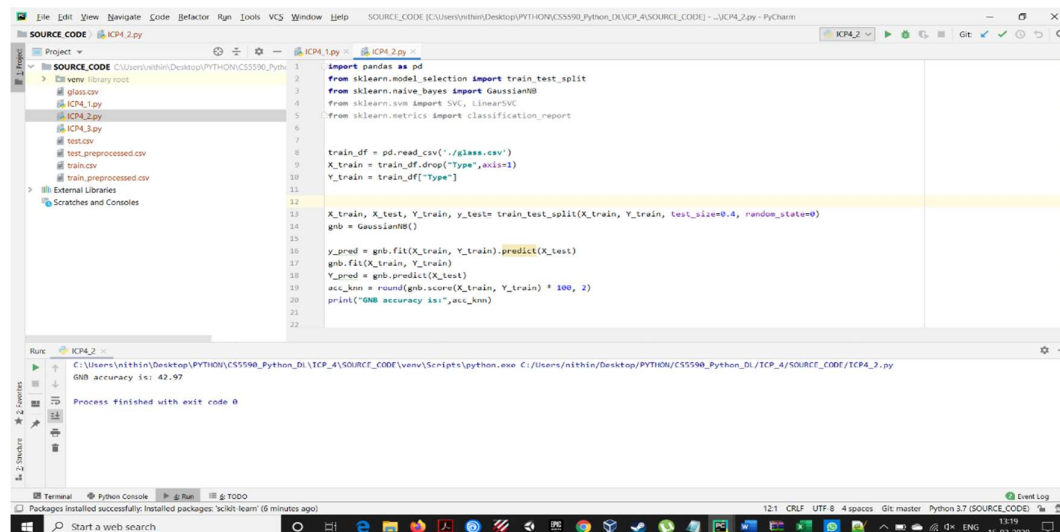
Run: ICP4_1

Sex	Survived
0	0.188908
1	0.742038

Process Finished with exit code 0

From the above we can see that there is more probability for females to survive rather than males. So, it is required to use the sex column to keep.

2. Implement Naïve Bayes method using scikit-learn library Use dataset available in <https://umkc.box.com/s/anjic6g6034ptm0hgii6fhcu919kx8x> Use train_test_split to create training and testing part Evaluate the model on testing part using score and classification_report(y_true, y_pred)



```
1 import pandas as pd
2 from sklearn.model_selection import train_test_split
3 from sklearn.naive_bayes import GaussianNB
4 from sklearn.svm import SVC, LinearSVC
5 from sklearn.metrics import classification_report
6
7 train_df = pd.read_csv("../glass.csv")
8 X_train = train_df.drop("Type", axis=1)
9 Y_train = train_df["Type"]
10
11
12 X_train, X_test, Y_train, Y_test = train_test_split(X_train, Y_train, test_size=0.4, random_state=0)
13 gnb = GaussianNB()
14
15 y_pred = gnb.fit(X_train, Y_train).predict(X_test)
16 gnb.fit(X_train, Y_train)
17 Y_pred = gnb.predict(X_test)
18 acc_knn = round(gnb.score(X_train, Y_train) * 100, 2)
19 print("GNB accuracy %s"%acc_knn)
20
21
22
```

Run: ICP4_2

GNB accuracy is: 42.97

Process Finished with exit code 0

3. Implement linear SVM method using scikit library
 Use the same dataset above
 Use `train_test_split` to create training and testing part
 Evaluate the model on testing part using `score` and `classification_report(y_true, y_pred)`
 Which algorithm you got better accuracy? Can you justify why?

```

1 import pandas as pd
2 from sklearn.model_selection import train_test_split
3 from sklearn.svm import SVC, LinearSVC
4 from sklearn.neighbors import KNeighborsClassifier
5 train_df = pd.read_csv("../glass.csv")
6 X_train = train_df.drop("Type", axis=1)
7 Y_train = train_df["Type"]
8
9
10 X_train, X_test, Y_train, Y_test = train_test_split(X_train, Y_train, test_size=0.4, random_state=0)
11 print(train_df[train_df.isnull().any(axis=1)])
12
13 #SVM
14 svc = SVC()
15 svc.fit(X_train, Y_train)
16 Y_pred = svc.predict(X_test)
17 acc_svc = round(svc.score(X_train, Y_train) * 100, 2)
18 print("svm accuracy is:", acc_svc)
  
```

Run: ICP4_3 ×
 C:\Users\nithin\Desktop\PYTHON\CS5590_Python_DL\ICP_4\SOURCE_CODE\venv\Scripts\python.exe C:\Users\nithin\Desktop\PYTHON\CS5590_Python_DL\ICP_4\SOURCE_CODE\ICP4_3.py
 Empty DataFrame
 Columns: [RI, Na, Mg, Al, Si, K, Ca, Ba, Fe, Type]
 Index: []
 svm accuracy is: 33.59
 Process finished with exit code 0

Naive Bayes algorithm has more accuracy than support vector machine algorithm. From a theoretical point of view, it is a little bit hard to compare the two methods. One is probabilistic in nature, while the second one is geometric. However, it's quite easy to come up with a function where one has dependencies between variables which are not captured by Naive Bayes ($y(a,b) = ab$), so we know it isn't an universal approximator. SVMs with the proper choice of Kernel are (as are 2/3-layer neural networks) though, so from that point of view, the theory matches the practice. But in the end, it comes down to performance on *your* problem - you basically want to choose the simplest method which will give good enough results for your problem *and* have a good enough performance. Spam detection has been famously solvable by just Naive Bayes, for example. Face recognition in images by a similar method enhanced with boosting etc.

Dukkipati, Sri Sai Nithin Chowdary

Class Id: 4

Team: 6