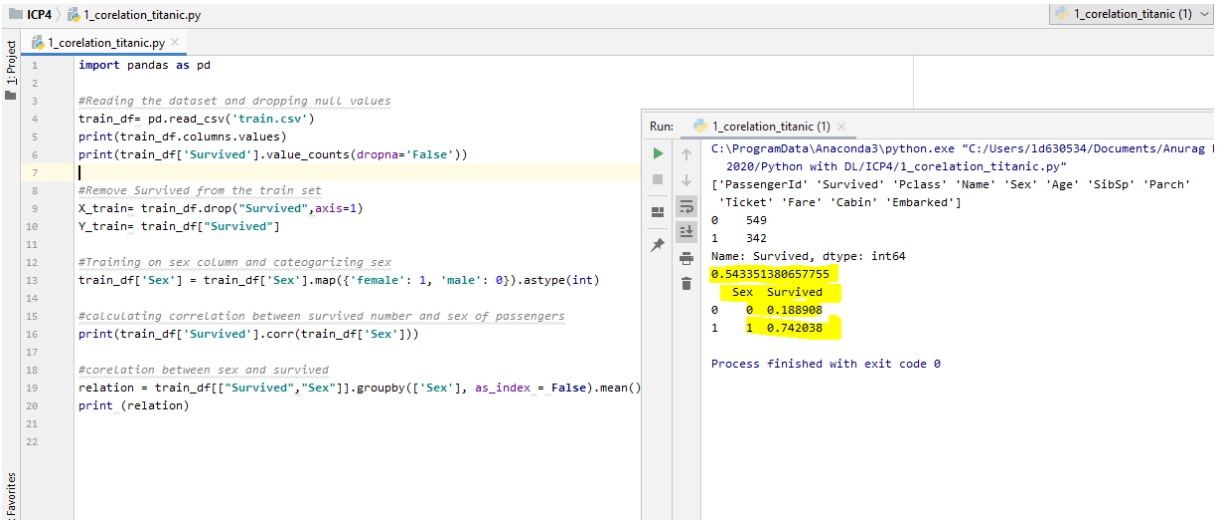
CS5590/490 - Python and Deep Learning Programming
In Class Programming Report - 4
Class ID 24 - Anurag Thantharate
Date Submitted: 02/16/2020

Video Link: https://www.loom.com/share/da1857f462c34d839be9a2c6b88656ca

---

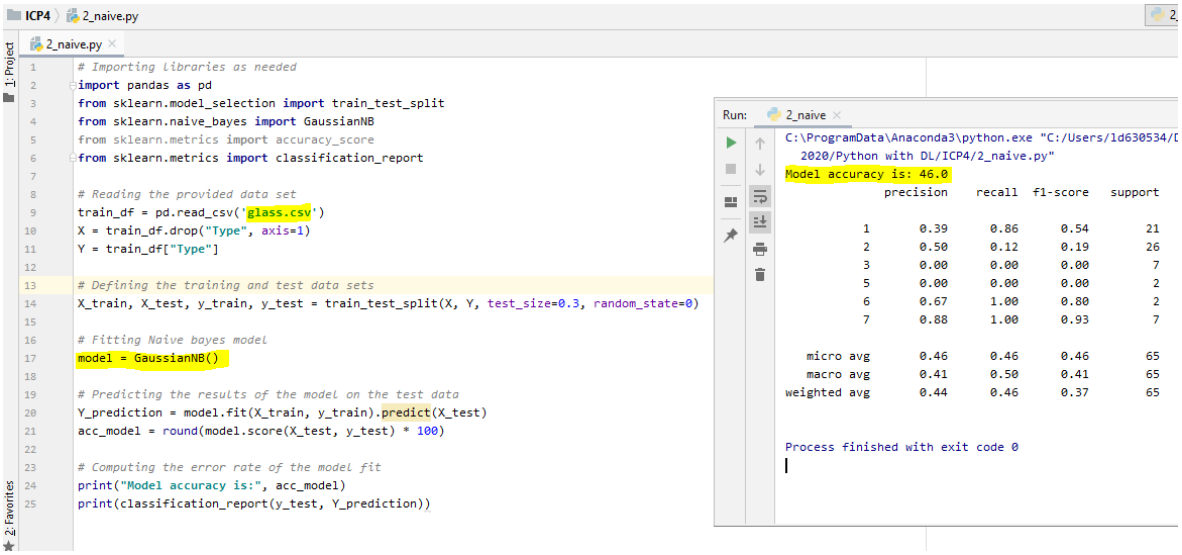
- Find the correlation between 'survived' (target column) and 'sex' column for the Titanic use case in class. Do you think we should keep this feature?



```
import pandas as pd

#Reading the dataset and dropping null values
train_df= pd.read_csv('train.csv')
print(train_df.columns.values)
print(train_df['Survived'].value_counts(dropna='False'))

#Remove Survived from the train set
X_train= train_df.drop("Survived",axis=1)
Y_train= train_df["Survived"]

#Training on sex column and cateogarizing sex
train_df['Sex'] = train_df['Sex'].map({'female': 1, 'male': 0}).astype(int)

#calculating correlation between survived number and sex of passengers
print(train_df['Survived'].corr(train_df['Sex']))

#corelation between sex and survived
relation = train_df[["Survived","Sex"]].groupby(['Sex'], as_index = False).mean()
print (relation)
```

```
C:\ProgramData\Anaconda3\python.exe "C:/Users/1d630534/Documents/Anurag
  2020/Python with DL/ICP4/1_corelation_titanic.py"
['PassengerId' 'Survived' 'Pclass' 'Name' 'Sex' 'Age' 'SibSp' 'Parch'
 'Ticket' 'Fare' 'Cabin' 'Embarked']
0    549
1    342
Name: Survived, dtype: int64
0.543351380657755
   Sex  Survived
0    0  0.188908
1    1  0.742038

Process finished with exit code 0
```

- Implement Naïve Bayes method using scikit-learn library.
  - Use dataset available in https://umkc.box.com/s/anji6c8g6034ptm0hgii6fhcu919kx8x
  - Use train_test_split to create training and testing part
  - Evaluate the model on testing part using score and



```
# Importing libraries as needed
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report

# Reading the provided data set
train_df = pd.read_csv('glass.csv')
X = train_df.drop("Type", axis=1)
Y = train_df["Type"]

# Defining the training and test data sets
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=0)

# Fitting Naive bayes model
model = GaussianNB()

# Predicting the results of the model on the test data
Y_prediction = model.fit(X_train, y_train).predict(X_test)
acc_model = round(model.score(X_test, y_test) * 100)

# Computing the error rate of the model fit
print("Model accuracy is:", acc_model)
print(classification_report(y_test, Y_prediction))
```

```
C:\ProgramData\Anaconda3\python.exe "C:/Users/1d630534/D
  2020/Python with DL/ICP4/2_naive.py"
Model accuracy is: 46.0
              precision    recall  f1-score   support

           1       0.39      0.86      0.54        21
           2       0.50      0.12      0.19        26
           3       0.00      0.00      0.00         7
           5       0.00      0.00      0.00         2
           6       0.67      1.00      0.80         2
           7       0.88      1.00      0.93         7

   micro avg       0.46      0.46      0.46        65
   macro avg       0.41      0.50      0.41        65
weighted avg       0.44      0.46      0.37        65


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

```python
# Importing libraries as needed
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.svm import SVC

# Reading the provided data set
# Defining the training and test
train_df = pd.read_csv('glass.csv')
X = train_df.drop("Type", axis=1)
Y = train_df["Type"]
 data sets
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=0)

# Fitting Naive bayes model
svc = SVC()
svc.fit(X_train, y_train)
Y_prediction= svc.predict(X_test)
acc_svc = round(svc.score(X_train, y_train) * 100,2)

# Computing the error rate of the model fit
print("SVM accuracy is:", acc_svc)
print(classification_report(y_test, Y_prediction))
```

```
Run: 3_SVM
  "avoid this warning.", FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\cla
 UndefinedMetricWarning: Precision and F-score are ill-define
 labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
SVM accuracy is: 77.18
              precision    recall  f1-score   support

           1       0.52      0.76      0.62        21
           2       0.70      0.62      0.65        26
           3       0.00      0.00      0.00         7
           5       1.00      1.00      1.00         2
           6       0.00      0.00      0.00         2
           7       0.88      1.00      0.93         7

   micro avg       0.63      0.63      0.63        65
   macro avg       0.51      0.56      0.53        65
weighted avg       0.57      0.63      0.59        65


Process finished with exit code 0
```

Which algorithm you got better accuracy? Can you justify why?

SVM got better accuracy in provided dataset. SVMs are more likely to perform better as they can handle non-linearities in the data. Naive Bayes performs best when the features are independent of each other.
