Problem- 1: [10 Points] To familiarize yourself with the environment, you will first go through the scenario of 'Titanic: Machine Learning from Disaster' and will build a model to predict the survival of passengers.

Pre-Processing:

On the first glance, it was clear that some attributes are redundant and will not make any significant contribution to my model, so I decided to drop them altogether. These were Ticket, PassengerId and Cabin. The values in the Cabin attribute were largely missing and the Ticket attribute contained a high number or duplicates. Also the Pclass and the Fare attribute made the Ticket attribute seem pointless. Lastly, the passengerId is just a primary-key attribute and also does not contribute to our model.

Next we took a look in the name category and extracted the titles from the name field. The most common titles were extracted and all the entries were categorized according to titles. The more rare titles were generalized into one category. After this, the name field was deleted. The misspelled titles were also fixed.

The Age attribute contained missing entries. This was fixed by replacing the missing ages with median of age of that particular Sex whose age was missing. After that, the Age field was banded and categorized by assigning an integer value according to range.

Then the Embarked field was fixed by replacing the missing values with the most popular value. The Parch and Sibsp field were removed in favor of a new attribute FamilySize which was the sum of the two attributes. Lastly, the fare attribute was banding and categorized according to ranges of fare.

Partitioning Data:

For my solution to this problem, I only considered the training data, as the 'ground truth' for the test dataset was not available. For this reason, I split the training dataset into a 60:40 ratio, where 60 % of the 892 total records where used to train the data and the remaining 40% was used to test it.

Results:

Model	Support Vector Machine	Logistical Regression	Naïve Bayes	Decision Trees
Accuracy	0.822	0.800	0.794	0.794
Precision	0.833	0.758	0.731	0.798
Recall	0.661	0.691	0.720	0.610
F -Measure	0.737	0.723	0.725	0.691

Conclusion:

All of the four classifiers performed well (scored out of 1) and consistently in all four measures they were scored against. Support Vector Machine outperformed the rest although by a very little margin which, I believe is not very significant enough to choose one over the other, hence other factors should be considered such as complexity and efficiency of the algorithm used. The results were consistent because I believe that there were not many attributes that were considered for the model in first place, as I disregarded the irrelevant ones form the start. Also the attributes that were given were well defined and homogenous (such as the Sex attribute), which led to such consistent results.

Reference:

https://www.kaggle.com/startupsci/titanic-data-science-solutions

Problem – 2: [40 Points] Choose any two of the given problems and build a predictive model for that problem using scikit-learn library.

SMS SPAM COLLECTION

Pre-Processing:

First I checked my Dataset for any immediate anomalies that could be rectified. The first problem I encountered was the hidden extra columns that were present in the data. I detected them using the describe() method in the panda library and dropped the extra columns.

Then I proceeded to convert the 'ham' and 'spam' values into 0's and 1's that indicated that the given SMS is a spam or not. After that I made my first attribute which was SMSLength. For this I calculated the SMSLength for each SMS and then calculated the mean of the spam messages using describe method. This Showed that the lengthiest spam text was of 225, the shortest was of 25, while the mean was of 80 characters. I did the same for the non-spam texts, which gave me a margin to work with. So I divided the SMSLength attribute into 3 categories corresponding to different SMSLength.

My second attribute was spamWordFreq which was a count of the number of commonly occurring spam words in a given text. For this, first I saved the 100 most commonly occurring words in both, spam and non-spam texts. Then I only kept the unique words that were in the spam words list and not in the non-spam common words list. Then I ran each text against this list and counted the number of occurrences of common spam words. This is a good measure as it was highly effective in highlighting the spam texts. I also divided this attribute into 4 categories. Finally I removed the text messages from my data set and ran the models.

Partitioning Data:

For my solution to this problem, I divided the data that was available into a 60:40 ratio. The data consisted of a total 5534 text messages but as texts vary greatly in words and content, I had to be generous with the test Data. For the 40 % that made up the test Data, I removed the solution and saved it into another file that was later used to score my model.

Results:

Model	Support Vector Machine	Logistical Regression	Naïve Bayes	Decision Trees
Accuracy	0.947	0.946	0.166	0.947
Precision	0.820	0.814	0.139	0.820
Recall	0.787	0.787	0.996	0.787
F -Measure	0.803	0.800	0.244	0.803

Conclusion:

Other than Naïve Bayes, all three classifiers performed really well. There weren't many attributes to begin with, so this was a rather simpler model to learn with only 24 different combinations of dataset. The attributes, though only two but were very homogeneous, and good indicators. I realized that Spam messages were mostly between a range of length and that turned out to be a good indicator. The low score of Naïve Bayes is interesting to note as other classifiers ran brilliantly, although I am clueless at the moment as to why this is the reason.

Problem – 2: [40 Points] Choose any two of the given problems and build a predictive model for that problem using scikit-learn library.

<u>VOICE RECONGNITION</u>

Pre-Processing:

First I had to shuffle the data as the data was not well defined. For that I used the built in function in sci kit library 'trainingData = shuffle(trainingData)'.

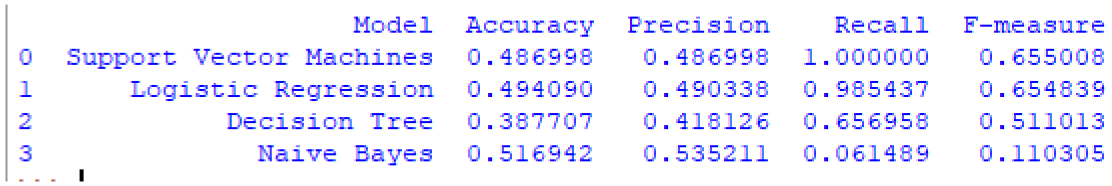
Then I saved the file as .csv and began to separate the data into testing and training sets.

Then I changed the labels from female and male, to 0s and 1s. For that I used a map and also changed the data type of the label field from object to string. After that I proceeded to drop the less important attributes, for this I used the describe() feature from the panda and compared the mean of each attribute in the training data set from the male and female set, and removed those features where the mean difference was not much. After that, I normalized the attributes from 1 so that the model could be learned easily. For this I also rounded the values to 2 decimal places.

Partitioning Data:

For my solution to this problem, I divided the data that was available into a 60:40 ratio. The data consisted of a total 3393 entries but as there were many attributes to learn, I had to be generous with the test Data. For the 40 % that made up the test Data, I removed the solution and saved it into another file that was later used to score my model.

Results:


```
                       Model  Accuracy  Precision    Recall  F-measure
0  Support Vector Machines  0.486998   0.486998  1.000000   0.655008
1      Logistic Regression  0.494090   0.490338  0.985437   0.654839
2            Decision Tree  0.387707   0.418126  0.656958   0.511013
3              Naive Bayes  0.516942   0.535211  0.061489   0.110305
```

Conclusion:

The score was less than optimal because I did not normalize the data very well. As the data was in float, the range of values were very large and hence the data lost its homogeneity. Also the rounded off feature was not appropriate here as all the attributes were rounded to the same decimal points regardless of there mean and standard deviation. A through statistical analysis is required to improve this model.