Data Analytics Questions

Assignment - 1

1)Commands for Performing simple linear regression (create linear regression model)?

```
regressor = LinearRegression() # Creating a regressor
regressor.fit(X_train,y_train) # Fiting the dataset into the model
```

We create a regressor. And we fit the X_train and y_train into the regressor model.

2) How to do Residual analysis(Check the results of model fitting to know whether the model is satisfactory)?

```
plt.scatter(X_test,y_test,color="green") # Plot a graph with X_test vs y_test
plt.plot(X_train,regressor.predict(X_train),color="red",linewidth=3) # Regressor line showing
plt.title('Regression(Test Set)')
plt.xlabel('HP')
plt.ylabel('MSRP')
plt.show()
```

3)How to do Predictions on the test set (apply the model)?

```
y_pred = regressor.predict(X_test)
print('R2 score: %.2f' % r2_score(y_test,y_pred)) # Priniting R 2 Score
print('Mean Error :',mean_squared_error(y_test,y_pred)) # Priniting the mean error
def car_price(hp): # A function to predict the price according to Horsepower
result = regressor.predict(np.array(hp).reshape(1, -1))
return(result[0,0])
car_hp = int(input('Enter Volkswagen cars HorsePower : '))
print('This Volkswagen Price will be : ',int(car_price(car_hp))*69,'₹')
```

4)write steps for Linear regression example with Python code and scikit-learn

1. First, let’s import linear_model from scikit-learn library:

```
from sklearn import linear_model
```

2. Now take features and labels set to train our program:

```
features = [[2],[1],[5],[10]]
labels = [27, 11, 75, 155]
```

3. After that create our model and fit the label and features to our model:

```
clf = linear_model.LinearRegression()
clf=clf.fit(features,labels)
```

4. In the end, pass data to the model and print the predicted result:

```
predicted = clf.predict([[8]])
print(predicted)
```

5)Give Steps to build Logistic Regression model in Python:

1. Import libraries /packages
2. Reading and understanding the data(do appropriate transformations- cleaning, filling nulls, duplicates, etc...)
3. Splitting our Data set in Dependent and Independent variables.

4. Performing simple linear regression (create logistic regression model)
5. Print the Accuracy and plot the Confusion Matrix
6. Print test data and predicted data Predictions on the test set

6)How to Split the data set into Dependent and Independent variables?

In our Data set we'll consider gmat score, gpa and Years of work_experience as Independent variable and admitted as Dependent Variable.

```
x = dataset.iloc[:, [2,3]].values
y = dataset.iloc[:, 4].values
```

Then, apply train_test_split. For example, you can set the test size to 0.25, and therefore the model testing
will be based on 25% of the dataset, while the model training will be based on 75% of the dataset:

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25,random_state=0)
```

7)How to do simple logistic regression (create a regression model)?

```
logistic_regression= LogisticRegression()
```



```
Mogallogistic_regression.fit(x_train,y_train)
y_pred=logistic_regression.predict(x_test)
```

8) Print the Accuracy and plot the Confusion Matrix

Then, use the code below to get the Confusion Matrix:

```
confusion_matrix = pd.crosstab(y_test, y_pred, rownames=['Actual'], colnames=['Predicted'])
sn.heatmap(confusion_matrix, annot=True)
```

For the final part, print the Accuracy and plot the Confusion Matrix:

```
print('Accuracy: ',metrics.accuracy_score(y_test, y_pred))
plt.show()
```

When we put all the code components together and Run the code in Python, will get the following Confusion
Matrix with an Accuracy of 0.8
(note that depending on sklearn version, may get different accuracy results. In above code case, the sklearn version is 0.22.2):
As can be observed from the matrix:
TP = True Positives = 4
TN = True Negatives = 4
FP = False Positives = 1
FN = False Negatives = 1
You can then also get the Accuracy using:
Accuracy = (TP+TN) / Total = (4+4) / 10 = 0.8
The accuracy is therefore 80% for the test set.

Assignment - 2

9) How Print test data and predicted data Predictions on the test set ?
Diving Deeper into the Results -> print two components in the python code:
print (x_test)
print (y_pred)

10) How to generate the association rules?
Generate association rules that have a support value of at least 5%
rules = association_rules(freq_items, metric ='support', min_threshold=0.05)
rules = rules.sort_values(['support', 'confidence'], ascending =[False,False])
print(rules)

11) How to Convert Pandas DataFrame into a list of lists for encoding?
transactions = [ ]
for i in range(0, len(df)):
transactions.append([str(df.values[i,j]) for j in range(0, len(df.columns))])

Assignment - 3
12) Natural Language Toolkit (NLTK) :
NLTK is a Python Package for performing Natural Language Processing on human language data which is mostly unstructured. It mainly focuses on analyzing textual data. It supports different natural language processing algorithms such as Tokenization, Frequency Distribution, Stopwords, Lexicon Normalization, Stemming, Lemmatization, POS Tagging. These are considered as pre-processing steps to perform text analytics.

13) Tokenization : It is the first step to perform text analytics. Tokenization means breaking down a textual paragraph into small chunks such as words or sentences. It is classified into two sections :
Sentence Tokenization and Word Tokenization : Sentence Tokenization breaks the text into sentences whereas Word Tokenization breaks the text into words.
Frequency Distribution :
The frequency distribution helps to understand how many words have occurred how many times in the given textual data.

14) Stopwords : Stopwords are considered as Noise in textual data. For example if text is containing words such as is, are, am, a, this, the, an etc. then they are treated as stopwords.

15) Stemming : Stemming is a process of linguistics normalization to reduce words to their word root or divide the derivational affixes. For example : writing, wrote, written can stemmed or reduced as write.

16) Lemmatization : Lemmatization is a process of removing words to their base words which are linguistically correct lemmas. For example : “Running” word will be lemmatized to “run”.