Slip 1,12,13,21

```
# Simple Linear Regression

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Salary_Data.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 1/3, random_state = 0)

# Training the Simple Linear Regression model on the Training set
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)

# Predicting the Test set results
y_pred = regressor.predict(X_test)

# Visualising the Training set results
plt.scatter(X_train, y_train, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.title('Salary vs Experience (Training set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()

# Visualising the Test set results
plt.scatter(X_test, y_test, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.title('Salary vs Experience (Test set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()
```

Slip 5,29

```
# Logistic Regression

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
```

```
dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
print(X_train)
print(y_train)
print(X_test)
print(y_test)

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
print(X_train)
print(X_test)

# Training the Logistic Regression model on the Training set
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)

# Predicting a new result
print(classifier.predict(sc.transform([[30,87000]])))

# Predicting the Test set results
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)

# Visualising the Training set results
from matplotlib.colors import ListedColormap
X_set, y_set = sc.inverse_transform(X_train), y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 10, stop = X_set[:, 0].max() + 10,
step = 0.25),
                     np.arange(start = X_set[:, 1].min() - 1000, stop = X_set[:, 1].max() + 1000, step
= 0.25))
plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(),
X2.ravel()]).T)).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
```

```
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('red', 'green'))(i),
label = j)
plt.title('Logistic Regression (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

Slip 6,11,15

Step 1: Install the libraries

```
pip install mlxtend
```

Step 2: Import the libraries

```
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules
```

Step 3: Read the data, encode the data

Create the sample dataset

```
transactions = [['eggs', 'milk','bread'],
['eggs', 'apple'],
['milk', 'bread'],
['apple', 'milk'],
['milk', 'apple', 'bread']]
from mlxtend.preprocessing import TransactionEncoder
te=TransactionEncoder()
te_array=te.fit(transactions).transform(transactions)
df=pd.DataFrame(te_array, columns=te.columns_)
df
freq_items = apriori(df, min_support = 0.5, use_colnames = True)
print(freq_items)
```

Step 5: 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)
```

Slip 7,8,30

```
# Apriori

# Run the following command in the terminal to install the apyori package: pip install apyori

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Data Preprocessing
dataset = pd.read_csv('Market_Basket_Optimisation.csv', header = None)
transactions = []
for i in range(0, 7501):
  transactions.append([str(dataset.values[i,j]) for j in range(0, 20)])

# Training the Apriori model on the dataset
from apyori import apriori
rules = apriori(transactions = transactions, min_support = 0.003, min_confidence = 0.2, min_lift = 3, min_length = 2, max_length = 2)

# Visualising the results

## Displaying the first results coming directly from the output of the apriori function
results = list(rules)
results

## Putting the results well organised into a Pandas DataFrame
def inspect(results):
    lhs         = [tuple(result[2][0][0])[0] for result in results]
    rhs         = [tuple(result[2][0][1])[0] for result in results]
    supports    = [result[1] for result in results]
    confidences = [result[2][0][2] for result in results]
    lifts       = [result[2][0][3] for result in results]
    return list(zip(lhs, rhs, supports, confidences, lifts))
resultsinDataFrame = pd.DataFrame(inspect(results), columns = ['Left Hand Side', 'Right Hand Side', 'Support', 'Confidence', 'Lift'])

## Displaying the results non sorted
resultsinDataFrame

## Displaying the results sorted by descending lifts
resultsinDataFrame.nlargest(n = 10, columns = 'Lift')
```

Slip 16,17

```
import nltk
nltk.download('all')
#Preprocessing
import re
text="""
Large paragraph of text
"""
text = re.sub(r'[[0-9]*]', ' ', text)
text = re.sub(r's+', ' ', text)
# Removing Square Brackets, digits, special symbols
import re
text = re.sub(r'[[0-9]{}*]', ' ', text)
# Removing special characters and digits
formatted_text = re.sub('[^a-zA-Z]', ' ', text)
```

Slip 18

```
from nltk.tokenize import sent_tokenize
from nltk.tokenize import word_tokenize
# Take some textual content to tokenize it in sentences and words.
paragraph_text="""Hello all, Welcome to Python Programming Academy. Python Programming Academy is a nice platform to learn new programming skills. It is difficult to get enrolled in this Academy."""
# Supply textual content to sent_tokenize() and word_tokenize()
tokenized_text_data=sent_tokenize(paragraph_text)
tokenized_words=word_tokenize(paragraph_text)
print("Tokenized Sentences : \n", tokenized_text_data, "\n")
print("Tokenized Words : \n",tokenized_words, "\n")

frequency_distribution=FreqDist(tokenized_words)
print(frequency_distribution)
```

Slip 18,20

```
import matplotlib.pyplot as plt
frequency_distribution.plot(32,cumulative=False)
plt.show()
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
# It will find the stowords in English language.
stop_words_data=set(stopwords.words("english"))
print(stop_words_data)
stop_words_data=set(stopwords.words("english"))
# Create a stopwords list to filter it from original text
filtered_words_list=[]
for words in tokenized_words:
if words not in stop_words_data:
filtered_words_list.append(words)
print("Tokenized Words : \n",tokenized_words,"\n")
print("Filtered Words : \n",filtered_words_list,"\n")
```