Sammed Jain

6.	Create a model to recognize hand written number images from the dataset (dataset is obtained by using load_digits function). Task1: Find the accuracy of the model after the training. Task2: Predict the number of first 10 digits of the test dataset. Task3: Draw a confusion matrix to know the situations when the prediction went wrong.	11-12
7.	Use cross validation technique to determine the best suited algorithm among Logistic regression, SVC, Random forest Classifier that could be implemented for recognizing digits from hand written number images (dataset is obtained by using load_digits function). Task1: Determine the best suited algorithm among the above. Task2: Perform parameter tunning to identify the attribute values that need to be provided to the algorithm (so that the accuracy rate will increase) which is determined as the best suited one from the above task ,using k fold cross validation.	13-14
8.	Train a model to detect whether a is a spam message or not using Multi Nomial Navie Bayes algorithm. (dataset is present in kaggle web application) Task1: Test the accuracy of the model after training Task2: Detect whether the following email message is spam or not. a)Hey Roy, can we get together to watch football game tomorrow? b)Upto 20% discount on parking, exclusive offer just for you. Don’t miss this reward!	15-16
9.	Use the following dataset to Task1: Draw elbow plot and from that figure out optimal value of k (k is an attribute used in KMeans algorithm) Task2: determine the number of clusters that could be formed using the k value obtained for the above dataset.	17-18

10.	Use the hand written image recognition dataset (dataset is obtained by using load_digits function). Task1: Train the model using Logistic Regression algorithm and print the accuracy of the model Task2: Apply Principle Component Analysis on the dataset and print the accuracy of the Logistic Regression model. Task3: Compare the accuracy of the model, before and after the use of PCA.	19-20

1. Create a Linear regression model using the following data.

area	price
2600	550000
3000	565000
3200	610000
3600	680000
4000	725000

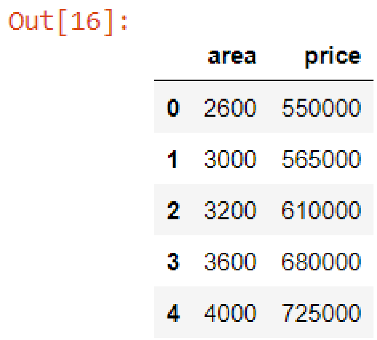
Task1: Draw a scatter plot using the data given above

Task2:train the model and predict price of a home with area = 3300 sqr ft

<u>Source Code:</u>

Task1:

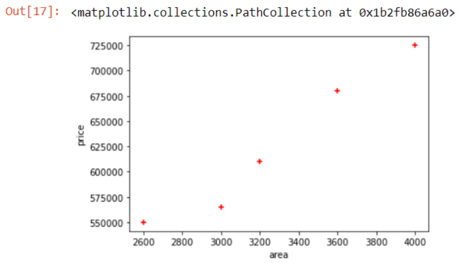
```
import pandas as pd

import numpy as np

from sklearn import linear_model

import matplotlib.pyplot as plt


df = pd.read_csv('D:\ML\lab\csv\program1.csv')

df
```



```
Out[16]:
      area   price
   0  2600  550000
   1  3000  565000
   2  3200  610000
   3  3600  680000
   4  4000  725000
```

<u>Task1:</u>

```
%matplotlib inline

plt.xlabel('area')

plt.ylabel('price')

plt.scatter(df.area,df.price,color='red',marker='+')
```

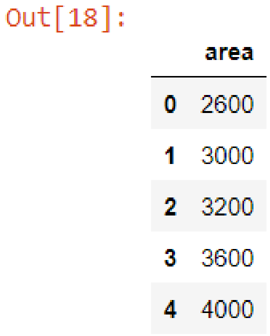


```
Out[17]: <matplotlib.collections.PathCollection at 0x1b2fb86a6a0>
```

Task2:

new_df = df.drop('price',axis='columns')

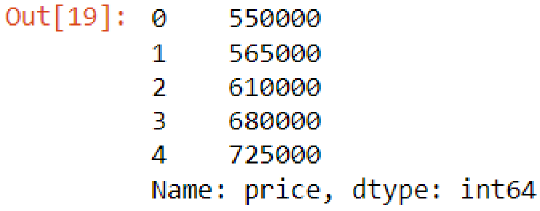
new_df



```
Out[18]:
            area
         0  2600
         1  3000
         2  3200
         3  3600
         4  4000
```

price = df.price

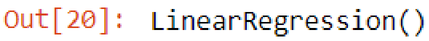
price



```
Out[19]: 0    550000
         1    565000
         2    610000
         3    680000
         4    725000
         Name: price, dtype: int64
```

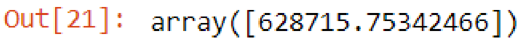
reg = linear_model.LinearRegression()

reg.fit(new_df,price)



```
Out[20]: LinearRegression()
```

reg.predict([[3300]])



```
Out[21]: array([628715.75342466])
```

2. Create a Logistic regression model to recognize hand written numbers.

Task1: use load_digits function to get the dataset of hand written numbers for recognition

Task2: train the model and test the accuracy of the model

Task3: Predict the first five hand written numbers in the dataset using the model

Task4: Draw a confusion matrix to know the situations when the prediction went wrong.

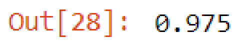
Source Code:

Task1:

```
from sklearn.datasets import load_digits
%matplotlib inline
import matplotlib.pyplot as plt
digits = load_digits()
```

Task2:

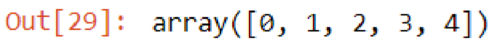
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(digits.data,digits.target,test_size=0.2)
model.fit(X_train, y_train)
model.score(X_test, y_test)
```



```
Out[28]: 0.975
```

Task3:

```
model.predict(digits.data[0:5])
```

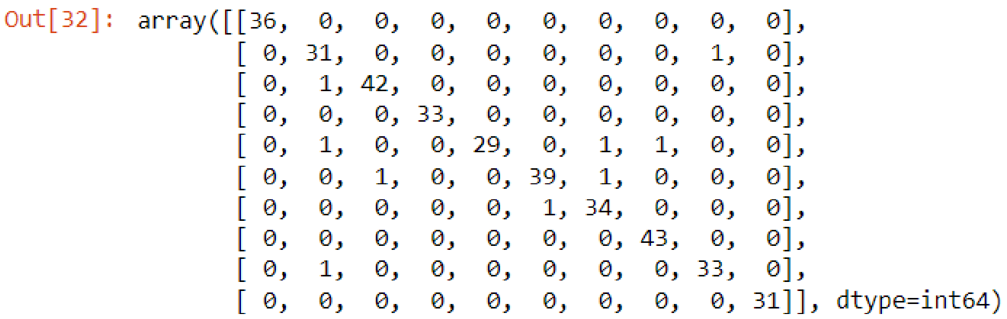


```
Out[29]: array([0, 1, 2, 3, 4])
```

Task4:

```
y_predicted = model.predict(X_test)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_predicted)
```

cm



```
Out[32]: array([[36,  0,  0,  0,  0,  0,  0,  0,  0,  0],
               [ 0, 31,  0,  0,  0,  0,  0,  0,  1,  0],
               [ 0,  1, 42,  0,  0,  0,  0,  0,  0,  0],
               [ 0,  0,  0, 33,  0,  0,  0,  0,  0,  0],
               [ 0,  1,  0,  0, 29,  0,  1,  1,  0,  0],
               [ 0,  0,  1,  0,  0, 39,  1,  0,  0,  0],
               [ 0,  0,  0,  0,  0,  1, 34,  0,  0,  0],
               [ 0,  0,  0,  0,  0,  0,  0, 43,  0,  0],
               [ 0,  1,  0,  0,  0,  0,  0,  0, 33,  0],
               [ 0,  0,  0,  0,  0,  0,  0,  0,  0, 31]], dtype=int64)
```

3. Train a linear regression model using the following dataset.

Mileage	Age	Sell Price
69000	6	18000
35000	3	34000
57000	5	26100
22500	2	40000
46000	4	31500

Task1: train a linear regression model and print the accuracy of the model

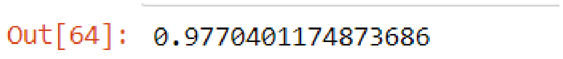
Task2: save the model into a file using ‘joblib’ library.

Task3: Load the saved file and print the predicted value for Mileage: 30000 , Age:4.

Source Code:

Task1:

```
import pandas as pd
import numpy as np
from sklearn import linear_model

df = pd.read_csv("D:\ML\lab\csv\program3.csv")

import matplotlib.pyplot as plt
%matplotlib inline

x = df[['Mileage','Age']]
y = df['Sell Price']

reg = linear_model.LinearRegression()
reg.fit(x,y)
reg.score(x,y)
```



```
Out[64]:  0.9770401174873686
```

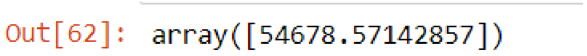
Task2:

```
from joblib import Parallel, delayed
```

```
import joblib
joblib.dump(reg, 'D:/ML/lab/filename.pkl')
```

Task3:

```
reg1 = joblib.load('D:/ML/lab/filename.pkl')
reg1.predict([[30000,4]])
```



```
Out[62]: array([54678.57142857])
```

4. Train a Decision tree model using the following dataset.

company	job	degree	salary_more_then_100k
google	sales executive	bachelors	0
google	sales executive	masters	0
google	business manager	bachelors	1
google	business manager	masters	1
google	computer programmer	bachelors	0
google	computer programmer	masters	1
abc pharma	sales executive	masters	0
abc pharma	computer programmer	bachelors	0
abc pharma	business manager	bachelors	0
abc pharma	business manager	masters	1
facebook	sales executive	bachelors	1
facebook	sales executive	masters	1
facebook	business manager	bachelors	1
facebook	business manager	masters	1
facebook	computer programmer	bachelors	1
facebook	computer programmer	masters	1

Task1: Train the model and print the accuracy of the model.

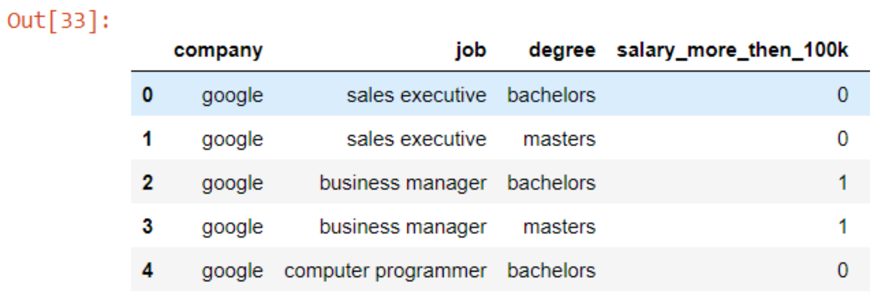
Task2: Print the predicted output if

Is salary of Google, Computer Engineer, Bachelors degree > 100 k ?

Source Code:

Task1:

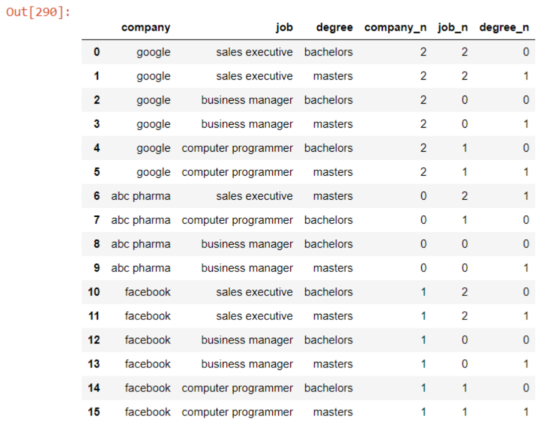
```
import pandas as pd

df = pd.read_csv("D:\ML\lab\csv\program4.csv")
df.head()
```



```
Out[33]:
     company                  job     degree  salary_more_then_100k
  0  google       sales executive  bachelors                      0
  1  google       sales executive    masters                      0
  2  google      business manager  bachelors                      1
  3  google      business manager    masters                      1
  4  google   computer programmer  bachelors                      0
```

```
inputs = df.drop('salary_more_then_100k',axis='columns')
target = df['salary_more_then_100k']
```

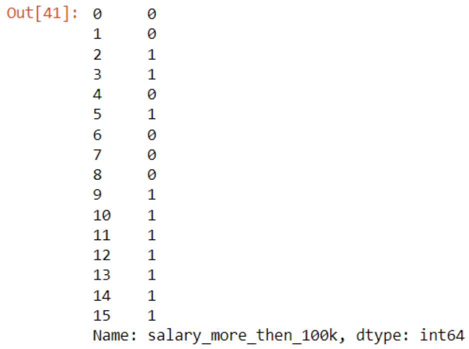
```
from sklearn.preprocessing import LabelEncoder
le_company = LabelEncoder()
le_job = LabelEncoder()
le_degree = LabelEncoder()

inputs['company_n'] = le_company.fit_transform(inputs['company'])
inputs['job_n'] = le_job.fit_transform(inputs['job'])
inputs['degree_n'] = le_degree.fit_transform(inputs['degree'])
inputs
```



Out[290]:

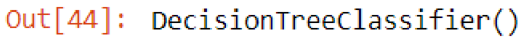
	company	job	degree	company_n	job_n	degree_n
0	google	sales executive	bachelors	2	2	0
1	google	sales executive	masters	2	2	1
2	google	business manager	bachelors	2	0	0
3	google	business manager	masters	2	0	1
4	google	computer programmer	bachelors	2	1	0
5	google	computer programmer	masters	2	1	1
6	abc pharma	sales executive	masters	0	2	1
7	abc pharma	computer programmer	bachelors	0	1	0
8	abc pharma	business manager	bachelors	0	0	0
9	abc pharma	business manager	masters	0	0	1
10	facebook	sales executive	bachelors	1	2	0
11	facebook	sales executive	masters	1	2	1
12	facebook	business manager	bachelors	1	0	0
13	facebook	business manager	masters	1	0	1
14	facebook	computer programmer	bachelors	1	1	0
15	facebook	computer programmer	masters	1	1	1

```
new_input = inputs.drop(['company','job','degree'],axis='columns')
target
```



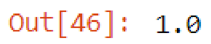
```
Out[41]: 0     0
         1     0
         2     1
         3     1
         4     0
         5     1
         6     0
         7     0
         8     0
         9     1
         10    1
         11    1
         12    1
         13    1
         14    1
         15    1
         Name: salary_more_then_100k, dtype: int64
```

```
from sklearn import tree
model = tree.DecisionTreeClassifier()
model.fit(new_input, target)
```



```
Out[44]: DecisionTreeClassifier()
```

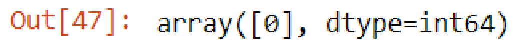
model.score(new_input,target)



```
Out[46]:  1.0
```

Task2:

model.predict([[2,1,0]])



```
Out[47]:  array([0], dtype=int64)
```

5. Train a SVM model using iris dataset (load the dataset using load_iris function).

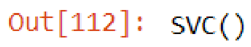
Task1: Train the model and print the accuracy of the model

Task2: Print the predicted output of the model when ‘sepal length’ = 4.8, ‘sepal width’=3.0, ‘petal length’=1.5, ‘petal width’=0.3

<u>Source Code:</u>

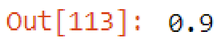
<u>Task1:</u>

```
import pandas as pd
from sklearn.datasets import load_iris
iris = load_iris()
df = pd.DataFrame(iris.data,columns=iris.feature_names)
df.head()

df['target'] = iris.target
df['flower_name'] =df.target.apply(lambda x: iris.target_names[x])
from sklearn.model_selection import train_test_split
X = df.drop(['target','flower_name'], axis='columns')
y = df.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
from sklearn.svm import SVC
model = SVC()
model.fit(X_train, y_train)
```



```
Out[112]:  SVC()
```

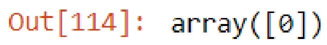
```
model.score(X_test, y_test)
```



```
Out[113]:  0.9
```

<u>Task2:</u>

```
model.predict([[4.8,3.0,1.5,0.3]])
```



```
Out[114]:  array([0])
```

6. Create a model to recognize hand written number images from the dataset (dataset is obtained by using load_digits function).

Task1: Find the accuracy of the model after the training.

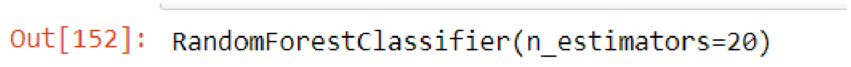
Task2: Predict the number of first 10 digits of the test dataset.

Task3: Draw a confusion matrix to know the situations when the prediction went wrong.

Source Code:

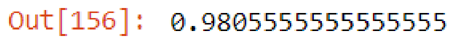
Task1:

```
import pandas as pd
from sklearn.datasets import load_digits
digits = load_digits()
df = pd.DataFrame(digits.data)
df.head()

df['target'] = digits.target

X = df.drop('target',axis='columns')
y = df.target

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2)

from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=20)
model.fit(X_train, y_train)
```



```
Out[152]: RandomForestClassifier(n_estimators=20)
```

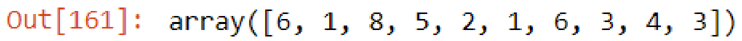
```
model.score(X_test, y_test)
```



```
Out[156]: 0.9805555555555555
```

Task2:

```
model.predict(X_test[:10])
```



```
Out[161]: array([6, 1, 8, 5, 2, 1, 6, 3, 4, 3])
```

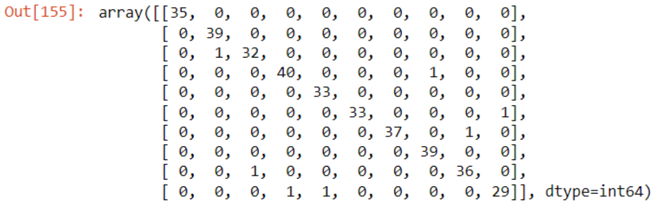
<u>Task3:</u>

y_predicted = model.predict(X_test)

from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_predicted)

cm



```
Out[155]: array([[35,  0,  0,  0,  0,  0,  0,  0,  0,  0],
                 [ 0, 39,  0,  0,  0,  0,  0,  0,  0,  0],
                 [ 0,  1, 32,  0,  0,  0,  0,  0,  0,  0],
                 [ 0,  0,  0, 40,  0,  0,  0,  1,  0,  0],
                 [ 0,  0,  0,  0, 33,  0,  0,  0,  0,  0],
                 [ 0,  0,  0,  0,  0, 33,  0,  0,  0,  1],
                 [ 0,  0,  0,  0,  0,  0, 37,  0,  1,  0],
                 [ 0,  0,  0,  0,  0,  0,  0, 39,  0,  0],
                 [ 0,  0,  1,  0,  0,  0,  0,  0, 36,  0],
                 [ 0,  0,  0,  1,  1,  0,  0,  0,  0, 29]], dtype=int64)
```

7. Use cross validation technique to determine the best suited algorithm among Logistic regression, SVC, Random forest Classifier that could be implemented for recognizing digits from hand written number images (dataset is obtained by using load_digits function).

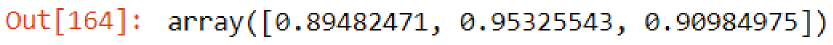
Task1: Determine the best suited algorithm among the above.

Task2: Perform parameter tunning to identify the attribute values that need to be provided to the algorithm (so that the accuracy rate will increase) which is determined as the best suited one from the above task ,using k fold cross validation.

Source Code:

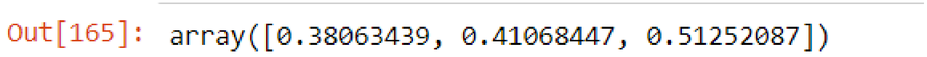
Task1:

```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
import numpy as np
from sklearn.datasets import load_digits
import matplotlib.pyplot as plt
digits = load_digits()
from sklearn.model_selection import cross_val_score

cross_val_score(LogisticRegression(solver='liblinear',multi_class='ovr'), digits.data,
digits.target,cv=3)
```



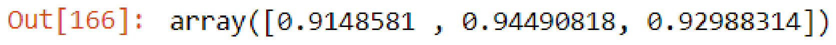
```
Out[164]: array([0.89482471, 0.95325543, 0.90984975])
```

```
cross_val_score(SVC(gamma='auto'), digits.data, digits.target,cv=3)
```



```
Out[165]: array([0.38063439, 0.41068447, 0.51252087])
```

```
cross_val_score(RandomForestClassifier(n_estimators=40),digits.data,
digits.target,cv=3)
```

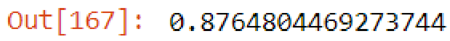


```
Out[166]: array([0.9148581 , 0.94490818, 0.92988314])
```

The algorithm that is best suited is Random forest classifier

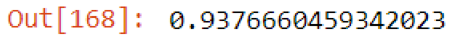
Task2:

```
scores1 = cross_val_score(RandomForestClassifier(n_estimators=5),digits.data, digits.target, cv=10)
np.average(scores1)
```



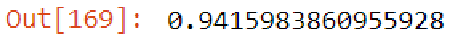
```
Out[167]: 0.8764804469273744
```

```
scores2 = cross_val_score(RandomForestClassifier(n_estimators=20),digits.data, digits.target, cv=10)
np.average(scores2)
```



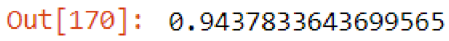
```
Out[168]: 0.9376660459342023
```

```
scores3 = cross_val_score(RandomForestClassifier(n_estimators=30),digits.data, digits.target, cv=10)
np.average(scores3)
```



```
Out[169]: 0.9415983860955928
```

```
scores4 = cross_val_score(RandomForestClassifier(n_estimators=40),digits.data, digits.target, cv=10)
np.average(scores4)
```



```
Out[170]: 0.9437833643699565
```

The attribute n_estimators with 40 as the value is the one that gives high accuracy to the model when passed into RandomForesrClassifier function.

8. Train a model to detect whether a is a spam message or not using Multi Nomial Navie Bayes algorithm.

(dataset is present in kaggle web application)

Task1: Test the accuracy of the model after training

Task2: Detect whether the following email message is spam or not.

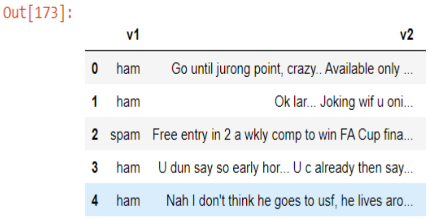
a)Hey Roy, can we get together to watch football game tomorrow?

b)Upto 20% discount on parking, exclusive offer just for you. Don’t miss this reward!

Source Code:

Task1:

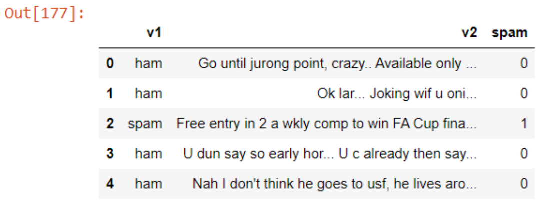
```
import pandas as pd
df = pd.read_csv("D:\ML\lab\csv\program8.csv")
df.head()
```



Out[173]:

	v1	v2
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

```
df['spam']=df['Category'].apply(lambda x: 1 if x=='spam' else 0)
df.head()
```



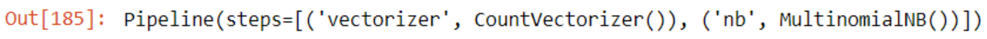
Out[177]:

	v1	v2	spam
0	ham	Go until jurong point, crazy.. Available only ...	0
1	ham	Ok lar... Joking wif u oni...	0
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	1
3	ham	U dun say so early hor... U c already then say...	0
4	ham	Nah I don't think he goes to usf, he lives aro...	0

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df.Message,df.spam)

from sklearn.feature_extraction.text import CountVectorizer
v = CountVectorizer()
from sklearn.naive_bayes import MultinomialNB
model = MultinomialNB()
```

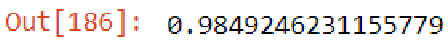
```
from sklearn.pipeline import Pipeline
clf = Pipeline([
    ('vectorizer', CountVectorizer()),
    ('nb', MultinomialNB())
])
```

```
clf.fit(X_train, y_train)
```



```
Out[185]: Pipeline(steps=[('vectorizer', CountVectorizer()), ('nb', MultinomialNB())])
```

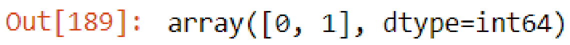
```
clf.score(X_test,y_test)
```



```
Out[186]: 0.9849246231155779
```

Task2:

```
emails = [
    'Hey Roy, can we get together to watch footbal game tomorrow?',
    'Upto 20% discount on parking, exclusive offer just for you. Dont miss this reward!'
]
clf.predict(emails)
```



```
Out[189]: array([0, 1], dtype=int64)
```

9. Use the following dataset to

Name	Age	Income($)
Rob	27	70000
Michael	29	90000
Mohan	29	61000
Ismail	28	60000
Kory	42	150000
Gautam	39	155000
David	41	160000
Andrea	38	162000
Brad	36	156000
Angelina	35	130000
Donald	37	137000
Tom	26	45000
Arnold	27	48000
Jared	28	51000
Stark	29	49500
Ranbir	32	53000
Dipika	40	65000
Priyanka	41	63000
Nick	43	64000
Alia	39	80000
Sid	41	82000
Abdul	39	58000

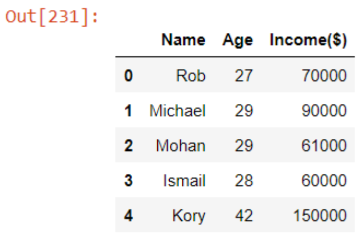
Task1: Draw elbow plot and from that figure out optimal value of k (k is an attribute used in KMeans algorithm)

Task2: determine the number of clusters that could be formed using the k value obtained for the above dataset.

Source Code:

Task1:

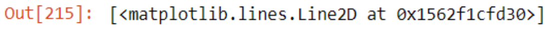
```
from sklearn.cluster import KMeans
import pandas as pd
from matplotlib import pyplot as plt
%matplotlib inline

df = pd.read_csv("D:\ML\lab\csv\program9.csv")
df.head()
```



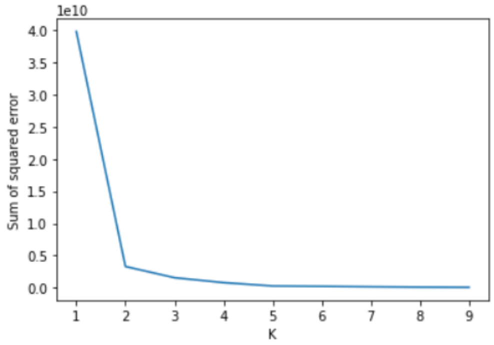
Out[231]:

	Name	Age	Income($)
0	Rob	27	70000
1	Michael	29	90000
2	Mohan	29	61000
3	Ismail	28	60000
4	Kory	42	150000

```
sse = []
k_rng = range(1,10)
for k in k_rng:
    km = KMeans(n_clusters=k)
    km.fit(df[['Age','Income($)']])
    sse.append(km.inertia_)
plt.xlabel('K')
plt.ylabel('Sum of squared error')
plt.plot(k_rng,sse)
```



```
Out[215]: [<matplotlib.lines.Line2D at 0x1562f1cfd30>]
```



Task2: Number of Clusters = ‘k’

So , Number of clusters that could be formed = 2 in the above case

10. Use the hand written image recognition dataset (dataset is obtained by using load_digits function).

Task1: Train the model using Logistic Regression algorithm and print the accuracy of the model

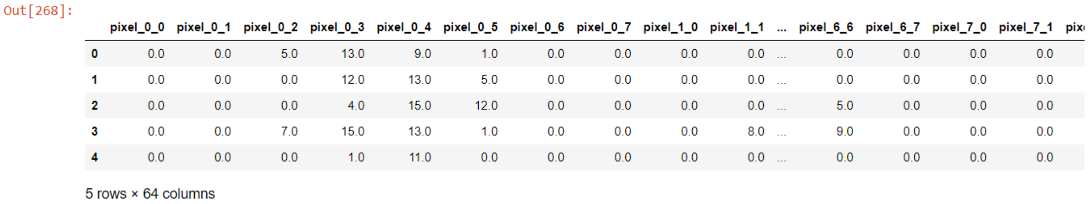
Task2: Apply Principle Component Analysis on the dataset and print the accuracy of the Logistic Regression model.

Task3: Compare the accuracy of the model, before and after the use of PCA.

Source Code:

Task1:

```
from sklearn.datasets import load_digits
import pandas as pd
dataset = load_digits()
df = pd.DataFrame(dataset.data, columns=dataset.feature_names)
df.head()
```

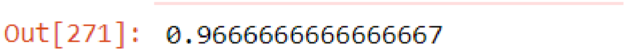


Out[268]:

	pixel_0_0	pixel_0_1	pixel_0_2	pixel_0_3	pixel_0_4	pixel_0_5	pixel_0_6	pixel_0_7	pixel_1_0	pixel_1_1	...	pixel_6_6	pixel_6_7	pixel_7_0	pixel_7_1	pix
0	0.0	0.0	5.0	13.0	9.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	12.0	13.0	5.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	4.0	15.0	12.0	0.0	0.0	0.0	0.0	...	5.0	0.0	0.0	0.0	
3	0.0	0.0	7.0	15.0	13.0	1.0	0.0	0.0	0.0	8.0	...	9.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	1.0	11.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	

5 rows × 64 columns

```
X = df
y = dataset.target
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=30)

from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, y_train)
model.score(X_test, y_test)
```

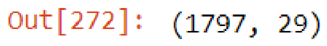


```
Out[271]: 0.9666666666666667
```

Task2:

```
from sklearn.decomposition import PCA
```

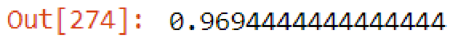
```
pca = PCA(0.95)
X_pca = pca.fit_transform(X)
X_pca.shape
```



```
Out[272]: (1797, 29)
```

```
X_train_pca, X_test_pca, y_train, y_test = train_test_split(X_pca, y, test_size=0.2, random_state=30)
```

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(max_iter=1000)
model.fit(X_train_pca, y_train)
model.score(X_test_pca, y_test)
```



```
Out[274]: 0.9694444444444444
```

Task3:The difference in the accuracy of the model ,before and after the use of PCA is not much