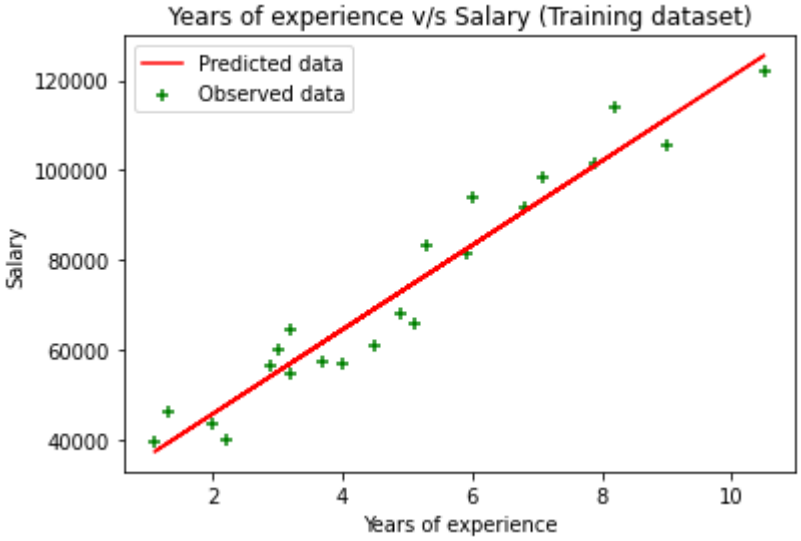
PRACTICAL 1 (SIMPLE LINEAR REGRESSION)

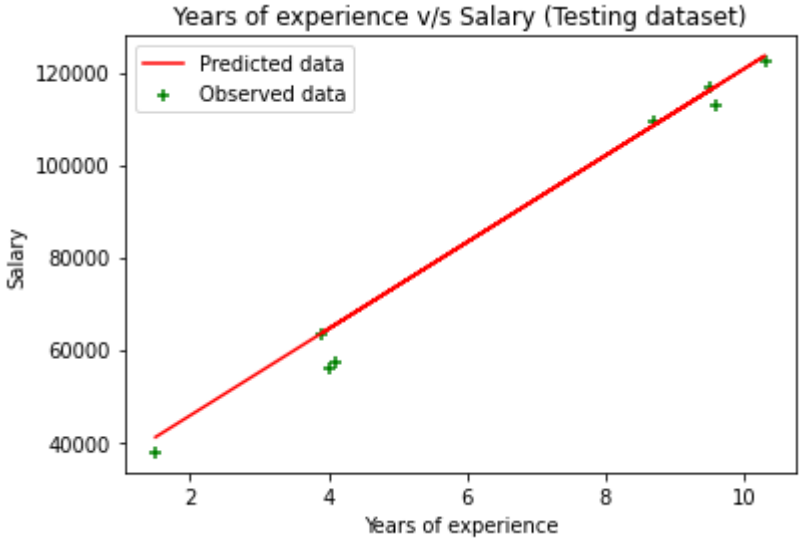
CODE AND OUTPUTS:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv("Salary_Data.csv")
X = dataset.iloc[ : ,  : -1].values
y = dataset.iloc[ : , -1].values

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 1/4,
random_state = 0)

from sklearn.linear_model import LinearRegression
linear_regression = LinearRegression()
linear_regression.fit(X_train, y_train)

y_train_pred = linear_regression.predict(X_train)
y_test_pred = linear_regression.predict(X_test)

plt.scatter(X_train, y_train, color = "green", marker = "+", label =
"Observed data")
plt.plot(X_train, y_train_pred, color = "red", label = "Predicted data")
plt.xlabel("Years of experience")
plt.ylabel("Salary")
plt.title("Years of experience v/s Salary (Training dataset)")
plt.legend()
plt.show()
```




```
plt.scatter(X_test, y_test, color = "green", marker = "+", label =
"Observed data")
plt.plot(X_test, y_test_pred, color = "red", label = "Predicted data")
plt.xlabel("Years of experience")
```

```
plt.ylabel("Salary")
plt.title("Years of experience v/s Salary (Testing dataset)")
plt.legend()
plt.show()
```



PRACTICAL 2 (MULTIPLE LINEAR REGRESSION)

CODE AND OUTPUTS:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

```
dataset = pd.read_csv('/content/50_Startups-2.csv')
x = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
```

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [3])],
remainder='passthrough')
x = np.array(ct.fit_transform(x))
```

```
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=0)
```

In [10]:

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(x_train, y_train)
```

```
y_pred = regressor.predict(x_test)
np.set_printoptions(precision=2)
print(np.concatenate((y_pred.reshape(len(y_pred),1),
y_test.reshape(len(y_test),1)),1))
```

```
[[103015.2  103282.38]
 [132582.28 144259.4 ]
 [132447.74 146121.95]
 [ 71976.1   77798.83]
 [178537.48 191050.39]
 [116161.24 105008.31]
 [ 67851.69  81229.06]
 [ 98791.73  97483.56]
 [113969.44 110352.25]
 [167921.07 166187.94]]
```

PRACTICAL 3 (SUPPORT VECTOR MACHINE)

CODE AND OUTPUTS:

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

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

```
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)
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

```
from sklearn.svm import SVC
classifier = SVC(kernel='linear', random_state=0)
classifier.fit(x_train, y_train)
```

```
print(classifier.predict(sc.transform([[30,200000]])))
```

```
[1]
```

In [20]:

```
y_pred = classifier.predict(x_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1),
y_test.reshape(len(y_test),1)),1))
```

```
[[0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [1 1]
```

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[66  2]
 [ 8 24]]
0.9
```

PRACTICAL 4 (KNN)

CODE AND OUTPUTS:

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

In [22]:

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

```
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)
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5, metric='minkowski', p=2)
classifier.fit(x_train, y_train)
```

Out[33]:

```
KNeighborsClassifier()
```

In [34]:

```
print(classifier.predict(sc.transform([[40, 200000]])))
```

```
[1]
```

```
y_pred = classifier.predict(x_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1),
y_test.reshape(len(y_test),1)),1))
```

```
[[0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [1 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]
 [1 0]
 [0 0]
 [0 0]
 [1 1]
```

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[64  4]
 [ 3 29]]
0.93
```

PRACTICAL 5 (HEIRARCHICAL CLUSTERING)

CODE AND OUTPUTS:

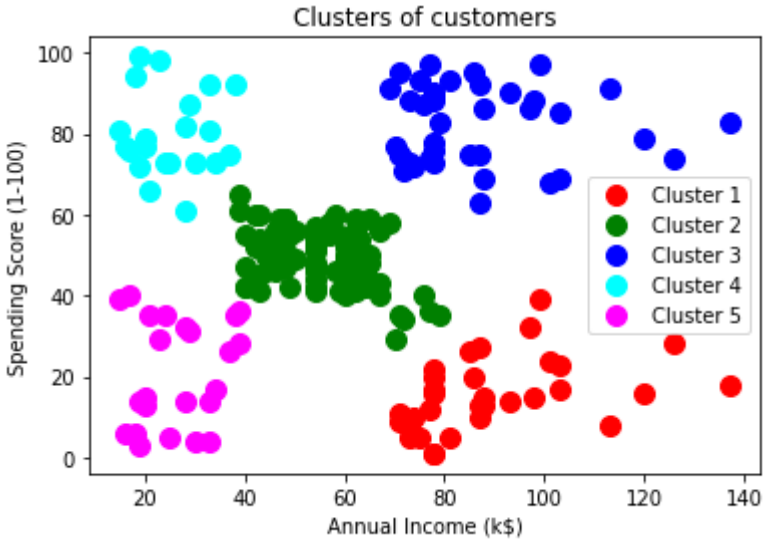
In [9]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
dataset = pd.read_csv('/content/Mall_Customers.csv')
X = dataset.iloc[:, [3,4]].values
```

```
from sklearn.cluster import AgglomerativeClustering
hc = AgglomerativeClustering(n_clusters=5, affinity='euclidean',
linkage='ward')
y_hc = hc.fit_predict(X)
```

```
plt.scatter(X[y_hc==0,0], X[y_hc==0,1], s=100, c='red', label='Cluster 1')
plt.scatter(X[y_hc==1,0], X[y_hc==1,1], s=100, c='green', label='Cluster
2')
plt.scatter(X[y_hc==2,0], X[y_hc==2,1], s=100, c='blue', label='Cluster 3')
plt.scatter(X[y_hc==3,0], X[y_hc==3,1], s=100, c='cyan', label='Cluster 4')
plt.scatter(X[y_hc==4,0], X[y_hc==4,1], s=100, c='magenta', label='Cluster
5')
plt.title('Clusters of customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```



PRACTICAL 6 (K MEANS CLUSTERING)

CODE AND OUTPUTS:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [2]:

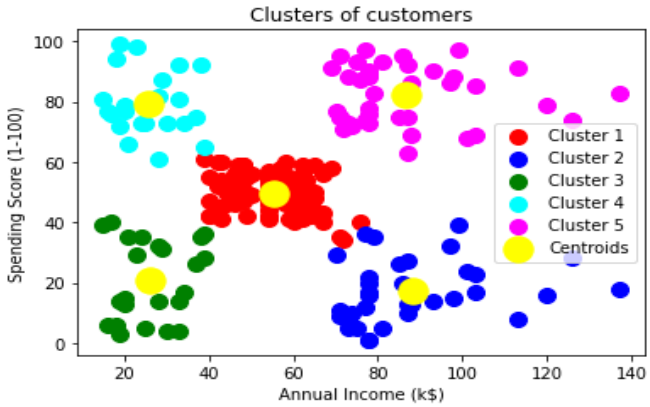
```
dataset = pd.read_csv('Mall_Customers.csv')
X = dataset.iloc[:, [3, 4]].values
```

In [3]:

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters = 5, init = 'k-means++', random_state = 42)
y_kmeans = kmeans.fit_predict(X)
print(y_kmeans)
```

```
[2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2
 3 2 3 2 3 2 0 2 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 4 1 4 0 4 1 4 1 4 0 4 1 4 1 4 1 4 1 4 0 4 1 4 1 4
 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1
 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4]
```

```
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 100, c = 'red',
label = 'Cluster 1')
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 100, c = 'blue',
label = 'Cluster 2')
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 100, c = 'green',
label = 'Cluster 3')
plt.scatter(X[y_kmeans == 3, 0], X[y_kmeans == 3, 1], s = 100, c = 'cyan',
label = 'Cluster 4')
plt.scatter(X[y_kmeans == 4, 0], X[y_kmeans == 4, 1], s = 100, c =
'magenta', label = 'Cluster 5')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s
= 300, c = 'yellow', label = 'Centroids')
plt.title('Clusters of customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```



PRACTICAL 7 (ANN)

CODE AND OUTPUTS:

```
import numpy as np
import pandas as pd
import tensorflow as tf
```

In [3]:

```
dataset = pd.read_csv('Churn_Modelling.csv')
X = dataset.iloc[:, 3:-1].values
y = dataset.iloc[:, -1].values
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
X[:, 2] = le.fit_transform(X[:, 2])
```

In [7]:

```
print(X)
```

```
[[619 'France' 0 ... 1 1 101348.88]
 [608 'Spain' 0 ... 0 1 112542.58]
 [502 'France' 0 ... 1 0 113931.57]
 ...
 [709 'France' 0 ... 0 1 42085.58]
 [772 'Germany' 1 ... 1 0 92888.52]
 [792 'France' 0 ... 1 0 38190.78]]
```

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [1])],
remainder='passthrough')
X = np.array(ct.fit_transform(X))
```

```
print(X)
```

```
[[1.0 0.0 0.0 ... 1 1 101348.88]
 [0.0 0.0 1.0 ... 0 1 112542.58]
 [1.0 0.0 0.0 ... 1 0 113931.57]
 ...
 [1.0 0.0 0.0 ... 0 1 42085.58]
 [0.0 1.0 0.0 ... 1 0 92888.52]
 [1.0 0.0 0.0 ... 1 0 38190.78]]
```

```
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 = 0)
```

In [11]:

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
```

```
X_test = sc.transform(X_test)
```

In [12]:

```
ann = tf.keras.models.Sequential()
```

In [13]:

```
ann.add(tf.keras.layers.Dense(units=6, activation='relu'))
```

In [14]:

```
ann.add(tf.keras.layers.Dense(units=6, activation='relu'))
```

In [15]:

```
ann.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))
```

In [16]:

```
ann.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics =
['accuracy'])
```

In [17]:

```
ann.fit(X_train, y_train, batch_size = 32, epochs = 100)
```

```
Epoch 1/100
250/250 [==============================] - 1s 1ms/step - loss: 0.5750 - accuracy: 0.7490
Epoch 2/100
250/250 [==============================] - 0s 1ms/step - loss: 0.4712 - accuracy: 0.7960
Epoch 3/100
250/250 [==============================] - 0s 2ms/step - loss: 0.4428 - accuracy: 0.7986
Epoch 4/100
250/250 [==============================] - 0s 2ms/step - loss: 0.4296 - accuracy: 0.8075
Epoch 5/100
250/250 [==============================] - 0s 2ms/step - loss: 0.4212 - accuracy: 0.8149
```

```
print(ann.predict(sc.transform([[1, 0, 0, 600, 1, 40, 3, 60000, 2, 1, 1,
50000]])) >0.5)
```

```
[[False]]
```

```
y_pred = ann.predict(X_test)
y_pred = (y_pred > 0.5)
print(np.concatenate((y_pred.reshape(len(y_pred),1),
y_test.reshape(len(y_test),1)),1))
```

```
[[0 0]
 [0 1]
 [0 0]
 ...
 [0 0]
 [0 0]
 [0 0]]
```

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[1499   96]
 [ 186  219]]
0.859
```

PRACTICAL 8 (CNN)

CODE AND OUTPUTS:

```
import tensorflow as tf
from keras.preprocessing.image import ImageDataGenerator
```

In [46]:

```
train_datagen = ImageDataGenerator(rescale=1./255, shear_range=0.2,
zoom_range=0.2, horizontal_flip=True)
training_set =
train_datagen.flow_from_directory('/content/drive/MyDrive/small_dataset/tra
ining_set', target_size=(64,64), batch_size=32, class_mode='binary')
```

```
Found 10 images belonging to 2 classes.
```

In [48]:

```
train_datagen = ImageDataGenerator(rescale=1./255, shear_range=0.2,
zoom_range=0.2, horizontal_flip=True)
test_set =
train_datagen.flow_from_directory('/content/drive/MyDrive/small_dataset/tes
t_set', target_size=(64,64), batch_size=32, class_mode='binary')
```

```
Found 10 images belonging to 2 classes.
```

In [49]:

```
cnn = tf.keras.models.Sequential()
```

In [50]:

```
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3,
activation='relu', input_shape=[64,64,3]))
```

In [51]:

```
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))
```

In [52]:

```
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3,
activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))
```

In [53]:

```
cnn.add(tf.keras.layers.Flatten())
```

```
cnn.add(tf.keras.layers.Dense(units=128, activation='relu'))
```

In [55]:

```
cnn.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))
```

In [56]:

```
cnn.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
```

In [57]:

```
cnn.fit(x=training_set, validation_data=test_set, epochs=25)
```

```
Epoch 1/25
1/1 [==============================] - 1s 897ms/step - loss: 0.7015 - accuracy: 0.5000 - val_loss: 0.7160 - val_accuracy: 0.5000
Epoch 2/25
1/1 [==============================] - 0s 227ms/step - loss: 0.6286 - accuracy: 0.9000 - val_loss: 0.7793 - val_accuracy: 0.5000
Epoch 3/25
1/1 [==============================] - 0s 224ms/step - loss: 0.6135 - accuracy: 0.5000 - val_loss: 0.7770 - val_accuracy: 0.5000
Epoch 4/25
1/1 [==============================] - 0s 230ms/step - loss: 0.6256 - accuracy: 0.4000 - val_loss: 0.7575 - val_accuracy: 0.4000
Epoch 5/25
1/1 [==============================] - 0s 211ms/step - loss: 0.5565 - accuracy: 0.9000 - val_loss: 0.7845 - val_accuracy: 0.4000
```

```
import numpy as np
from keras.preprocessing import image
test_image=image.load_img('/content/drive/MyDrive/small_dataset/single_pred
iction/cat_or_dog_1.jpg', target_size=(64,64))
test_image=image.img_to_array(test_image)
test_image=np.expand_dims(test_image, axis=0)
result=cnn.predict(test_image)
training_set.class_indices
if result[0][0]==1:
  prediction='dog'
else:
  prediction='cat'
```

In [63]:

```
print(prediction)
```

```
dog
```