In[1]:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline

data = pd.read_csv("Classified Data",index_col=0)
data.head()
```

In[2]:

```
from sklearn.preprocessing import StandardScaler
SS = StandardScaler()
SS.fit(df.drop('TARGET CLASS', axis = 1))
SS_features = SS.transform(df.drop('TARGET CLASS', axis = 1))
df_features = pd.DataFrame(SS_features, columns = df.columns[:-1])
df_features.head()
```

In[3]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(SS_features, df['TARGET CLASS'], test_size = 0.30)

from sklearn.neighbors import KNeighborsClassifier
```

```
model = KNeighborsClassifier(n_neighbors = 1)

model.fit(X_train,y_train)

predictions = model.predict(X_test)
```

In[4]:

```
from sklearn.metrics import accuracy_score

print(accuracy_score(y_test, pred))
```

In[1]:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

data = pd.read_csv("USA_Housing.csv")

data.head()
```

In[2]:

```
X = data[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms', 'Avg. Area Number of Bedrooms', 'Area Population']]

y = data['Price']
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.40)


from sklearn.linear_model import LinearRegression

model = LinearRegression()

model.fit(X_train, y_train)


predictions = model.predict(X_test)
```

In[3]:

```
from sklearn import metrics

print('MSE:', metrics.mean_squared_error(y_test, predictions))
```

In[1]:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline


data = pd.read_csv(“titanic_train.csv”)

data.head()
```

In[2]:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split ( data.drop('Survived',axis = 1), data['Survived'],
test_size = 0.3, random_state = 101)


from sklearn.linear_model import LogisticRegression

model = LogisticRegression()

model.fit(X_train, y_train)


predictions = model.predict(X_test)
```

In[3]:

```
from sklearn.metrics import classification_report

print(classification_report(y_test, predictions))
```

In[1]:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline
```

```
data = pd.read_csv("kyphosis.csv")
data.head()
```

In[2]:

```
from sklearn.model_selection import train_test_split

X = data.drop("Kyphosis", axis=1)
y =  data["Kyphosis"]

X_train, X_test, y_train, y_test = train_test_split (X, y, test_size = 0.3, random_state=101)

from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
model.fit(X_train, y_train)

predictions = model.predict(X_test)
```

In[3]:

```
from sklearn.metrics import classification_report, confusion_matrix

print(classification_report(y_test, predictions))
print(confusion_matrix(y_test, predictions))
```

In[4]:

```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(n_estimators = 250)

model.fit(X_train, y_train)


RFC_predictions = model.predict(X_test)


print(confusion_matrix(y_test, RFC_predictions))

print(classification_report(y_test, RFC_predictions))
```

In[1]:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline


data = pd.read_csv("loan_data.csv")

data.head()
```

In[2]:

```
Categorical_features = ['purpose']


new_data = pd.get_dummies(data, columns = cat_feats, drop_first = True)
```

In[3]:

```
from sklearn.model_selection import train_test_split

X = new_data.drop("not.fully.paid", axis=1)
y =  new_data["not.fully.paid"]

X_train, X_test, y_train, y_test = train_test_split (X, y, test_size = 0.3, random_state=101)
```

In[4]:

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators = 650)
model.fit(X_train, y_train)

RFC_predictions = model.predict(X_test)

from sklearn.metrics import classification_report, confusion_matrix

print(classification_report(y_test, RFC_predictions))
print(confusion_matrix(y_test, RFC_predictions))
```