**1. Write a Python program to prepare Scatter Plot for Iris Dataset**

%matplotlib inline

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

from sklearn.datasets import load\_iris

iris = load\_iris()

df= pd.DataFrame(data= np.c\_[iris['data'], iris['target']],

                 columns= iris['feature\_names'] + ['target'])

# select setosa and versicolor

y = df.iloc[0:100, 4].values

y = np.where(y == 'Iris-setosa', 0, 1)

# extract sepal length and petal length

X = df.iloc[0:100, [0, 2]].values

# plot data

plt.scatter(X[:50, 0], X[:50, 1],

            color='blue', marker='o', label='Setosa')

plt.scatter(X[50:100, 0], X[50:100, 1],

            color='green', marker='s', label='Versicolor')

plt.xlabel('Sepal length [cm]')

plt.ylabel('Petal length [cm]')

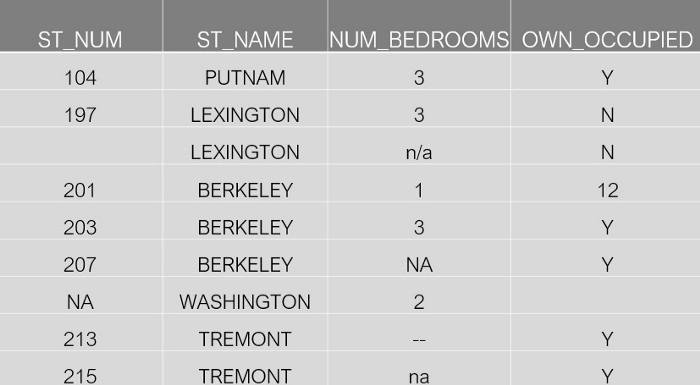
plt.legend(loc='upper left')

# plt.savefig('images/02\_06.png', dpi=300)

plt.show()

**2. Write a python program to find all null values in a given dataset and remove them.**

Firstly Create CSV File



# Importing libraries  
import pandas as pd  
import numpy as np  
  
# Read csv file into a pandas dataframe  
df = pd.read\_csv("property data.csv")  
  
# Take a look at the first few rows  
print df.head()

# Looking at the ST\_NUM column  
print df['ST\_NUM']  
print df['ST\_NUM'].isnull()

# Looking at the NUM\_BEDROOMS column  
print df['NUM\_BEDROOMS']  
print df['NUM\_BEDROOMS'].isnull()

# Making a list of missing value types  
missing\_values = ["n/a", "na", "--"]  
df = pd.read\_csv("property data.csv", na\_values = missing\_values)

# Looking at the OWN\_OCCUPIED column  
print df['OWN\_OCCUPIED']  
print df['OWN\_OCCUPIED'].isnull()

# Total missing values for each feature  
print df.isnull().sum()

# Replace missing values with a number  
df['ST\_NUM'].fillna(125, inplace=True)

**3. Write a python program to make Categorical values in numeric format for a given dataset**

# importing pandas as pd

import pandas as pd

# importing data using .read\_csv() function

df = pd.read\_csv('data.csv')

# printing DataFrame

df

# using .get\_dummies function to convert

# the categorical datatype to numerical

# and storing the returned dataFrame

# in a new variable df1

df1 = pd.get\_dummies(df['Purchased'])

# using pd.concat to concatenate the dataframes

# df and df1 and storing the concatenated

# dataFrame in df.

df = pd.concat([df, df1], axis=1).reindex(df.index)

# removing the column 'Purchased' from df

# as it is of no use now.

df.drop('Purchased', axis=1, inplace=True)

# printing df

df

**4.Write a python program to Implement Simple Linear Regression for predicting house price.**

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

HouseDF = pd.read\_csv('USA\_Housing.csv')

HouseDF.head()

HouseDF.info()

X = HouseDF[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',

'Avg. Area Number of Bedrooms', 'Area Population']]

y = HouseDF['Price']

**from** **sklearn.model\_selection** **import** train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.4, random\_state=101)

**from** **sklearn.linear\_model** **import** LinearRegression

lm = LinearRegression()

lm.fit(X\_train,y\_train)

print(lm.intercept\_)

coeff\_df = pd.DataFrame(lm.coef\_,X.columns,columns=['Coefficient']) coeff\_df

predictions = lm.predict(X\_test)

plt.scatter(y\_test,predictions)

sns.distplot((y\_test-predictions),bins=50);

**5.Write a python program to implement Multiple Linear Regression for given dataset.**

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

**#Reading the dataset**

dataset = pd.read\_csv("advertising.csv")

dataset.head()

**#Setting the value for X and Y**x = dataset[['TV', 'Radio', 'Newspaper']]  
y = dataset['Sales']

**#Splitting the dataset**from sklearn.model\_selection import train\_test\_splitx\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size = 0.3, random\_state = 100)

**#Fitting the Multiple Linear Regression model**mlr = LinearRegression()   
mlr.fit(x\_train, y\_train)

**#Intercept and Coefficient**

print("Intercept: ", mlr.intercept\_)  
print("Coefficients:")  
list(zip(x, mlr.coef\_))

**#Prediction of test set**

y\_pred\_mlr= mlr.predict(x\_test)

**#Predicted values**

print("Prediction for test set: {}".format(y\_pred\_mlr))

**#Actual value and the predicted value**

mlr\_diff = pd.DataFrame({'Actual value': y\_test, 'Predicted value': y\_pred\_mlr})  
slr\_diff.head()

**#Model Evaluation**

from sklearn import metrics

meanAbErr = metrics.mean\_absolute\_error(y\_test, y\_pred\_mlr)  
meanSqErr = metrics.mean\_squared\_error(y\_test, y\_pred\_mlr)  
rootMeanSqErr = np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred\_mlr))

print('R squared: {:.2f}'.format(mlr.score(x,y)\*100))  
print('Mean Absolute Error:', meanAbErr)  
print('Mean Square Error:', meanSqErr)  
print('Root Mean Square Error:', rootMeanSqErr)

**6. Write a python program to implement Polynomial Linear Regression for given dataset**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('Position\_Salaries.csv')

dataset

X = dataset.iloc[:,1:2].values

y = dataset.iloc[:,2].values

# fitting the linear regression model

from sklearn.linear\_model import LinearRegression

lin\_reg = LinearRegression()

lin\_reg.fit(X,y)

# visualising the linear regression model

plt.scatter(X,y, color='red')

plt.plot(X, lin\_reg.predict(X),color='blue')

plt.title("Truth or Bluff(Linear)")

plt.xlabel('Position level')

plt.ylabel('Salary')

plt.show()

# polynomial regression model

from sklearn.preprocessing import PolynomialFeatures

poly\_reg = PolynomialFeatures(degree=2)

X\_poly = poly\_reg.fit\_transform(X)

X\_poly     # prints X\_poly

lin\_reg2 = LinearRegression()

lin\_reg2.fit(X\_poly,y)

# visualising polynomial regression

from sklearn.preprocessing import PolynomialFeatures

poly\_reg = PolynomialFeatures(degree=4)

X\_poly = poly\_reg.fit\_transform(X)

lin\_reg2 = LinearRegression()

lin\_reg2.fit(X\_poly,y)

X\_grid = np.arange(min(X),max(X),0.1)

X\_grid = X\_grid.reshape(len(X\_grid),1)

plt.scatter(X,y, color='red')

plt.plot(X\_grid, lin\_reg2.predict(poly\_reg.fit\_transform(X\_grid)),color='blue')

plt.title("Truth or Bluff(Polynomial)")

plt.xlabel('Position level')

plt.ylabel('Salary')

plt.show()

7. Write a python program to implement Naive Bayes.

%matplotlib inline

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns; sns.set()

from sklearn.datasets import make\_blobs

X, y = make\_blobs(100, 2, centers=2, random\_state=2, cluster\_std=1.5)

plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='RdBu');

from sklearn.naive\_bayes import GaussianNB

model = GaussianNB()

model.fit(X, y);

rng = np.random.RandomState(0)

Xnew = [-6, -14] + [14, 18] \* rng.rand(2000, 2)

ynew = model.predict(Xnew)

plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='RdBu')

lim = plt.axis()

plt.scatter(Xnew[:, 0], Xnew[:, 1], c=ynew, s=20, cmap='RdBu', alpha=0.1)

plt.axis(lim);

yprob = model.predict\_proba(Xnew)

yprob[-8:].round(2)

**8. Write a python program to implement Decision Tree whether or not to play Tennis**

*# Load libraries*

**import** numpy **as** np

**import** pandas **as** pd

**from** sklearn **import** metrics *#Import scikit-learn metrics module for accuracy calculation*

len(df) *#Dataset Lenght*

df**.**shape *#To see the number of rows and columns in our dataset:*

df**.**head() *#To inspect the first five records of the dataset:*

df**.**describe() *#To see statistical details of the dataset:*

*#machine learning algorithms can only learn from numbers (int, float, doubles .. )*

*#so let us encode it to int*

**from** sklearn **import** preprocessing

string\_to\_int**=** preprocessing**.**LabelEncoder() *#encode your data*

df**=**df**.**apply(string\_to\_int**.**fit\_transform) *#fit and transform it*

df

*#To divide our data into attribute set and Label:*

feature\_cols **=** ['Outlook','Temprature','Humidity','Wind']

X **=** df[feature\_cols ] *#contains the attribute*

y **=** df**.**Play\_Tennis *#contains the label*

*#To divide our data into training and test sets:*

**from** sklearn.model\_selection **import** train\_test\_split

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X, y, test\_size**=**0.30)

*# perform training*

**from** sklearn.tree **import** DecisionTreeClassifier *# import the classifier*

classifier **=**DecisionTreeClassifier(criterion**=**"entropy", random\_state**=**100) *# create a classifier object*

classifier**.**fit(X\_train, y\_train) *# fit the classifier with X and Y d*

DecisionTreeClassifier(class\_weight=None, criterion='entropy', max\_depth=None,

max\_features=None, max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0, min\_impurity\_split=None,

min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0, presort=False, random\_state=100,

splitter='best')

*#Predict the response for test dataset*

y\_pred**=** classifier**.**predict(X\_test)

*# Model Accuracy, how often is the classifier correct?*

**from** sklearn.metrics **import** accuracy\_score

print("Accuracy:",metrics**.**accuracy\_score(y\_test, y\_pred))

data\_p**=**pd**.**DataFrame({'Actual':y\_test, 'Predicted':y\_pred})

data\_p

**from** sklearn.metrics **import** classification\_report, confusion\_matrix

print(confusion\_matrix(y\_test, y\_pred))

print(classification\_report(y\_test, y\_pred))

**from** sklearn.tree **import** export\_graphviz

**from** sklearn.externals.six **import** StringIO

**from** IPython.display **import** Image

**import** pydotplus

dot\_data **=** StringIO()

export\_graphviz(classifier, out\_file**=**dot\_data,

filled**=True**, rounded**=True**,

special\_characters**=True**,feature\_names **=**value,class\_names**=**['0','1'])

graph **=** pydotplus**.**graph\_from\_dot\_data(dot\_data**.**getvalue())

graph**.**write\_png('Play Tennis.png')

Image(graph**.**create\_png())

**9. Write a python program to implement Linear SVM.**

import numpy as np

import matplotlib.pyplot as plt import pandas as pd

dataset = pd.read\_csv('Social\_Network\_Ads.csv')

X = dataset.iloc[:, [2, 3]].values y = dataset.iloc[:, 4].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 = 'rbf', random\_state = 0) classifier.fit(X\_train, y\_train)

y\_pred = classifier.predict(X\_test)

from sklearn.metrics import confusion\_matrix, accuracy\_score cm = confusion\_matrix(y\_test, y\_pred) print(cm) accuracy\_score(y\_test,y\_pred)

from matplotlib.colors import ListedColormap

X\_set, y\_set = X\_test, y\_test

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01), np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(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('SVM (Test set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()