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

ST_NUM	ST_NAME	NUM_BEDROOMS	OWN_OCCUPIED
104	PUTNAM	3	Y
197	LEXINGTON	3	N
	LEXINGTON	n/a	N
201	BERKELEY	1	12
203	BERKELEY	3	Y
207	BERKELEY	NA	Y
NA	WASHINGTON	2	
213	TREMONT	--	Y
215	TREMONT	na	Y

```
# 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 houseprice.

```
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 librariesimport 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 Yx = dataset[['TV', 'Radio',
'Newspaper']]
y = dataset['Sales']


#Splitting the datasetfrom 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 modelmlr =
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()
```