1	Write a python program to Prepare Scatter Plot (Use Forge Dataset / Iris Dataset)
2	Write a python program the Categorical values in numeric format for a given dataset.
3	Write a python program to implement simple Linear Regression for predicting house price.
4	Write a python program to implement Polynomial Regression for given dataset
5	Write a python program to Implement Naïve Bayes.
6	Write a python program to Implement Decision Tree whether or not to play tennis.
7	Write a python program to implement linear SVM.
8	Write a python program to implement Agglomerative clustering on a synthetic dataset. Data Sets for ML – UCI Machine Learning Repository – www.kaggle.com

Assnmt 1: Write a python program to Prepare Scatter Plot (Use Forge Dataset / Iris Dataset)

```
import pandas as pd

import matplotlib.pyplot as plt

iris = pd.read_csv("Iris1.csv")

iris.head()

iris.plot(kind="scatter", x="SepalLengthCm", y="SepalWidthCm")
```

Assnmt 2: Write a python program the 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('DataMLcategorical2.csv')

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()


# Using .fit_transform function to fit label

# encoder and return encoded label
```

```
label = le.fit_transform(df['Purchased'])
# printing label
Label
# removing the column 'Purchased' from df
# as it is of no use now.
df.drop("Purchased", axis=1, inplace=True)

# Appending the array to our dataFrame
# with column name 'Purchased'
df["Purchased"] = label
 # printing Dataframe
df
```

Assnmt 3: Write a python program to implement simple Linear Regression for predicting salary.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
dataset = pd.read_csv('Salary_DataSimpleLinearRegression.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 = 1/3, random_state = 0)
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
y_pred = regressor.predict(X_test)
plt.scatter(X_train, y_train, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.title('Salary vs Experience (Training set)')

plt.xlabel('Years of Experience')

plt.ylabel('Salary')

plt.show()
```

Assnmt 4: Write a python program to implement Polynomial Regression for given dataset.

```
import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read_csv('Position_SalariesPolynomialRegression.csv')

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

y = dataset.iloc[:, -1].values

from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression()

lin_reg.fit(X, y)

from sklearn.preprocessing import PolynomialFeatures

poly_reg = PolynomialFeatures(degree = 4)

X_poly = poly_reg.fit_transform(X)

lin_reg_2 = LinearRegression()

lin_reg_2.fit(X_poly, y)

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

plt.plot(X, lin_reg_2.predict(poly_reg.fit_transform(X)), color = 'blue')

plt.title('Truth or Bluff Polynomial Regression')

plt.xlabel('Position level')

plt.ylabel('Salary')

plt.show()

lin_reg_2.predict(poly_reg.fit_transform([[6.5]]))
```

Assnmt 5: Write a python program to Implement Naïve Bayes.-No need of dataset

```
from sklearn.datasets import  load iris

iris = load_iris()

 # store the feature matrix (X) and response vector (y)

X = iris.data

y = iris.target

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=1)

from sklearn.naive_bayes import GaussianNB

gnb = GaussianNB()

gnb.fit(X_train, y_train)

y_pred = gnb.predict(X_test)

 from sklearn import metrics

print("Gaussian Naive Bayes model accuracy(in %):", metrics.accuracy_score(y_test, y_pred)*100)
```

Assnmt 6: Write a python program to implement linear SVM.Datset-Iris1.csv

```
import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

from sklearn import svm, datasets

iris = pd.read_csv('Iris1.csv')

iris = datasets.load_iris()

X = iris.data[:, :2]

y = iris.target

C = 1.0

svc = svm.SVC(kernel ='linear', C = 1).fit(X, y)

x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1

y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1

h = (x_max / x_min)/100
```

```
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),np.arange(y_min, y_max, h))

plt.subplot(1, 1, 1)

Z = svc.predict(np.c_[xx.ravel(), yy.ravel()])

Z = Z.reshape(xx.shape)

plt.contourf(xx, yy, Z, cmap = plt.cm.Paired, alpha = 0.8)

plt.scatter(X[:, 0], X[:, 1], c = y, cmap = plt.cm.Paired)

plt.xlabel('Sepal length')

plt.ylabel('Sepal width')

plt.xlim(xx.min(), xx.max())

plt.title('SVC with linear kernel')

plt.show()
```

Assnmt 7: Write a python program to Implement Decision Tree for a dataset.-Iris1.csv

```
# Python program to implement decision tree algorithm and plot the tree

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn import metrics

import seaborn as sns

from sklearn.datasets import load_iris

from sklearn.model_selection import train_test_split

from sklearn import tree

 iris = load_iris()

data = pd.DataFrame(data = iris.data, columns = iris.feature_names)

 data['Species'] = iris.target

target = np.unique(iris.target)

target_n = np.unique(iris.target_names)

target_dict = dict(zip(target, target_n))
```

```
data['Species'] = data['Species'].replace(target_dict)

x = data.drop(columns = "Species")

y = data["Species"]

names_features = x.columns

target_labels = y.unique()

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state = 93)

from sklearn.tree import DecisionTreeClassifier

dtc = DecisionTreeClassifier(max_depth = 3, random_state = 93)

dtc.fit(x_train, y_train)

plt.figure(figsize = (30, 10), facecolor = 'b')

Tree = tree.plot_tree(dtc, feature_names = names_features, class_names = target_labels, rounded = True, filled = True, fontsize = 14)

plt.show()

y_pred = dtc.predict(x_test)
```

Assnmt 8: Write a python program to implement *Agglomerative clustering* on a dataset.

```
import numpy as nm

import matplotlib.pyplot as mtp

import pandas as pd

dataset = pd.read_csv("ML_8_Mall_Customers.csv")

x = dataset.iloc[:, [3, 4]].values

import scipy.cluster.hierarchy as shc

dendro = shc.dendrogram(shc.linkage(x, method="ward"))

mtp.title("Dendrogrma Plot")

mtp.ylabel("Euclidean Distances")

mtp.xlabel("Customers")

mtp.show()

from sklearn.cluster import AgglomerativeClustering

hc= AgglomerativeClustering(n_clusters=5, affinity='euclidean', linkage='ward')
```

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