Machine Learning Assignment 1

Name: Abhishek Salwan

Course: COE

Roll No: 101503007

Batch: COE1

PART I

[For Regression Dataset | filename: regressionDataSet.csv]

Q1.1 Compare the performance of 10 machine learning models for given regression data set for thedata partition of 70-30% with acceptable error of ±100.

Table 1.1: Comparative Performance Study of Machine Learning Models

Model	Method	Package	r	R^2	Error	Accuracy
M1	Linear	sklearn.linear_model	1.0	1.0	0.0000001	100.0
M2	Polynomial	sklearn.linear_model	0.9999	0.9999	0.0009	100.0
M3	Lasso	sklearn.linear_model	0.9999	0.9998	11.175	100.0
M4	Elastic Net	sklearn.linear_model	0.9999	0.9999	3.7032	100.0
M5	Ridge	sklearn.linear_model	0.9999	0.9999	0.000004	100.0
M6	Bayesian Ridge	sklearn.linear_model	0.9999	0.9999	0.0474	100.0
M7	Kernel Ridge	sklearn.kernel_ridge	0.9999	0.9999	0.000004	100.0
M8	K-Neighbors	sklearn.neighbors	0.9988	0.9976	41.8680	94.4281
M9	Decision Tree	sklearn.tree	0.9981	0.9962	50.611	86.133
M10	Random Forest	sklearn.ensemble	0.9990	0.9980	40.0374	98.0310

M1 - Linear Regression Model

```
#importing the libraries
```

```
import pandas as pd
import math as m
import numpy as np
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error

#importing the dataset
dataset = pd.read_csv('regressionDataSet.csv')
x = dataset.iloc[:, 1:].values
y = dataset.iloc[:, 0].values

#splitting the dataset into training set and test set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 1/5)

#fitting the model on the training set
from sklearn.linear_model import LinearRegression
regressor=LinearRegression()
regressor.fit(x_train,y_train)

#predicting the test set results
y_pred=regressor.predict(x_test)

#calculating r2
r2=r2_score(y_test,y_pred)

#calculating r
r=m.sqrt(r2)

#calculating error
error=mean_absolute_error(y_test,y_pred)

#calculating accuracy
accuracy = (float)(np.count_nonzero(np.array(abs(y_test - y_pred) <=
100))/np.size(y_test))*100
```

M2 - Polynomimal Regression Model

```
#importing the libraries
import pandas as pd
import math as m
import numpy as np
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error

#importing the dataset
dataset = pd.read_csv('regressionDataSet.csv')
x = dataset.iloc[:, 1:].values
y = dataset.iloc[:, 0].values

#splitting the dataset into training set and test set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 1/5)

#fitting the model on the training set
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
poly_reg=PolynomialFeatures(degree=4)
x_poly=poly_reg.fit_transform(x_train)
regressor=LinearRegression()
regressor.fit(x_poly,y_train)

#predicting the test set results
y_pred=regressor.predict(poly_reg.fit_transform(x_test))

#calculating r2
r2=r2_score(y_test,y_pred)

#calculating r
r=m.sqrt(r2)

#calculating error
error=mean_absolute_error(y_test,y_pred)

#calculating accuracy
accuracy = (float)(np.count_nonzero(np.array(abs(y_test - y_pred) <=
100))/np.size(y_test))*100
```

M3 - Lasso Regression Model

```
#importing the libraries
import pandas as pd
import math as m
import numpy as np
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error

#importing the dataset
dataset = pd.read_csv('regressionDataSet.csv')
x = dataset.iloc[:, 1:].values
y = dataset.iloc[:, 0].values

#splitting the dataset into training set and test set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 1/5)

#fitting the model on the training set
from sklearn.linear_model import Lasso
regressor=Lasso(max_iter=10000)
regressor.fit(x_train,y_train)

#predicting the test set results
y_pred=regressor.predict(x_test)

#calculating r2
r2=r2_score(y_test,y_pred)

#calculating r
r=m.sqrt(r2)

#calculating error
error=mean_absolute_error(y_test,y_pred)
```

```
#calculating accuracy
accuracy = (float)(np.count_nonzero(np.array(abs(y_test - y_pred) <=
100))/np.size(y_test))*100
```

M4 - Elastic Net Regression Model

```
#importing the libraries
import pandas as pd
import math as m
import numpy as np
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error

#importing the dataset
dataset = pd.read_csv('regressionDataSet.csv')
x = dataset.iloc[:, 1:].values
y = dataset.iloc[:, 0].values

#splitting the dataset into training set and test set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 1/5)

#fitting the model on the training set
from sklearn.linear_model import ElasticNet
regressor=ElasticNet(max_iter=10000)
regressor.fit(x_train,y_train)

#predicting the test set results
y_pred=regressor.predict(x_test)

#calculating r2
r2=r2_score(y_test,y_pred)

#calculating r
r=m.sqrt(r2)

#calculating error
error=mean_absolute_error(y_test,y_pred)

#calculating accuracy
accuracy = (float)(np.count_nonzero(np.array(abs(y_test - y_pred) <=
100))/np.size(y_test))*100
```

M5 - Ridge Regression Model

```
#importing the libraries
import pandas as pd
import math as m
import numpy as np
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error

#importing the dataset
dataset = pd.read_csv('regressionDataSet.csv')
```

```
x = dataset.iloc[:, 1:].values
y = dataset.iloc[:, 0].values

#splitting the dataset into training set and test set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 1/5)

#fitting the model on the training set
from sklearn.linear_model import Ridge
regressor=Ridge()
regressor.fit(x_train,y_train)

#predicting the test set results
y_pred=regressor.predict(x_test)

#calculating r2
r2=r2_score(y_test,y_pred)

#calculating r
r=m.sqrt(r2)

#calculating error
error=mean_absolute_error(y_test,y_pred)

#calculating accuracy
accuracy = (float)(np.count_nonzero(np.array(abs(y_test - y_pred) <=
100))/np.size(y_test))*100
```

M6 - Bayesian Ridge Regression Model

```
#importing the libraries
import pandas as pd
import math as m
import numpy as np
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error

#importing the dataset
dataset = pd.read_csv('regressionDataSet.csv')
x = dataset.iloc[:, 1:].values
y = dataset.iloc[:, 0].values

#splitting the dataset into training set and test set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 1/5)

#fitting the model on the training set
from sklearn.linear_model import BayesianRidge
regressor=BayesianRidge()
regressor.fit(x_train,y_train)

#predicting the test set results
y_pred=regressor.predict(x_test)

#calculating r2
r2=r2_score(y_test,y_pred)
```

```
#calculating r
r=m.sqrt(r2)

#calculating error
error=mean_absolute_error(y_test,y_pred)

#calculating accuracy
accuracy = (float)(np.count_nonzero(np.array(abs(y_test - y_pred) <=
100))/np.size(y_test))*100
```

M7 - Kernel Ridge Regression Model

```
#importing the libraries
import pandas as pd
import math as m
import numpy as np
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error

#importing the dataset
dataset = pd.read_csv('regressionDataSet.csv')
x = dataset.iloc[:, 1:].values
y = dataset.iloc[:, 0].values

#splitting the dataset into training set and test set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 1/5)

#fitting the model on the training set
from sklearn.kernel_ridge import KernelRidge
regressor=KernelRidge()
regressor.fit(x_train,y_train)

#predicting the test set results
y_pred=regressor.predict(x_test)

#calculating r2
r2=r2_score(y_test,y_pred)

#calculating r
r=m.sqrt(r2)

#calculating error
error=mean_absolute_error(y_test,y_pred)

#calculating accuracy
accuracy = (float)(np.count_nonzero(np.array(abs(y_test - y_pred) <=
100))/np.size(y_test))*100
```

M8 - K-Neighbors Regression Model

```
#importing the libraries
import pandas as pd
import math as m
import numpy as np
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error

#importing the dataset
dataset = pd.read_csv('regressionDataSet.csv')
x = dataset.iloc[:, 1:].values
y = dataset.iloc[:, 0].values

#splitting the dataset into training set and test set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 1/5)

#fitting the model on the training set
from sklearn.neighbors import KNeighborsRegressor
regressor=KNeighborsRegressor(n_neighbors=3)
regressor.fit(x_train,y_train)

#predicting the test set results
y_pred=regressor.predict(x_test)

#calculating r2
r2=r2_score(y_test,y_pred)

#calculating r
r=m.sqrt(r2)

#calculating error
error=mean_absolute_error(y_test,y_pred)

#calculating accuracy
accuracy = (float)(np.count_nonzero(np.array(abs(y_test - y_pred) <=
100))/np.size(y_test))*100
```

M9 - Decision Tree Regression Model

```
#importing the libraries
import pandas as pd
import math as m
import numpy as np
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error

#importing the dataset
dataset = pd.read_csv('regressionDataSet.csv')
x = dataset.iloc[:, 1:].values
y = dataset.iloc[:, 0].values

#splitting the dataset into training set and test set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 1/5)

#fitting the model on the training set
from sklearn.tree import DecisionTreeRegressor
regressor=DecisionTreeRegressor(criterion='mae')
regressor.fit(x_train,y_train)
```

```
#predicting the test set results
y_pred=regressor.predict(x_test)

#calculating r2
r2=r2_score(y_test,y_pred)

#calculating r
r=m.sqrt(r2)

#calculating error
error=mean_absolute_error(y_test,y_pred)

#calculating accuracy
accuracy = (float)(np.count_nonzero(np.array(abs(y_test - y_pred) <=
100))/np.size(y_test))*100
```

M10 - Random Forest Regression Model

```
#importing the libraries
import pandas as pd
import math as m
import numpy as np
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error

#importing the dataset
dataset = pd.read_csv('regressionDataSet.csv')
x = dataset.iloc[:, 1:].values
y = dataset.iloc[:, 0].values

#splitting the dataset into training set and test set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 1/5)

#fitting the model on the training set
from sklearn.ensemble import RandomForestRegressor
regressor=RandomForestRegressor(n_estimators=500,random_state=0)
regressor.fit(x_train,y_train)

#predicting the test set results
y_pred=regressor.predict(x_test)

#calculating r2
r2=r2_score(y_test,y_pred)

#calculating r
r=m.sqrt(r2)

#calculating error
error=mean_absolute_error(y_test,y_pred)

#calculating accuracy
accuracy = (float)(np.count_nonzero(np.array(abs(y_test - y_pred) <=
100))/np.size(y_test))*100
```

Q1.2 Ensemble the models from Table 1.1 for data partition for given regression data set of 70-30%and with acceptable error of ±100.

Table 1.2: Result analysis of ensemble models

Model	Combination	r	R^2	Error	Accuracy
E1	M1,M5,M6,M7,M10	0.9999	0.9999	7.8300	100.0
E2	M1,M2,M4,M9,M10	0.9998	0.9996	16.7008	100.0
E3	M2,M4,M6,M8,M10	0.9998	0.9996	15.9994	100.0
E4	M1,M3,M5,M7	0.9999	0.9999	0.000003	100.0
E5	M1,M2,M6,M8,M10	0.9998	0.9996	16.4602	100.0

E1 - M1,M5,M6,M7,M10

```
#importing the libraries
import pandas as pd
import math as m
import numpy as np
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error

#importing the dataset
dataset = pd.read_csv('regressionDataSet.csv')
x = dataset.iloc[:, 1:].values
y = dataset.iloc[:, 0].values

#splitting the dataset into training set and test set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 1/5)

df=pd.DataFrame()

#fitting the linear model
from sklearn.linear_model import LinearRegression
regressor1=LinearRegression()
regressor1.fit(x_train,y_train)
y_pred1=regressor1.predict(x_test)
df['pred1']=y_pred1

#fitting the ridge model
from sklearn.linear_model import Ridge
regressor2=Ridge()
regressor2.fit(x_train,y_train)
y_pred2=regressor2.predict(x_test)
df['pred2']=y_pred2

#fitting the bayesian ridge model
from sklearn.linear_model import BayesianRidge
regressor3=BayesianRidge()
regressor3.fit(x_train,y_train)
```

```
y_pred3=regressor3.predict(x_test)
df['pred3']=y_pred3

#fitting the kernel ridge model
from sklearn.kernel_ridge import KernelRidge
regressor4=KernelRidge()
regressor4.fit(x_train,y_train)
y_pred4=regressor4.predict(x_test)
df['pred4']=y_pred4

#fitting the random forest model
from sklearn.ensemble import RandomForestRegressor
regressor5=RandomForestRegressor(n_estimators=500,random_state=0)
regressor5.fit(x_train,y_train)
y_pred5=regressor5.predict(x_test)
df['pred5']=y_pred5

#ensembling
y_pred=df.mean(axis=1)

#calculating r2
r2=r2_score(y_test,y_pred)

#calculating r
r=m.sqrt(r2)

#calculating error
error=mean_absolute_error(y_test,y_pred)

#calculating accuracy
accuracy = (float)(np.count_nonzero(np.array(abs(y_test - y_pred) <=
100))/np.size(y_test))*100
```

E2 - M1,M2,M4,M9,M10

```
#importing the libraries
import pandas as pd
import math as m
import numpy as np
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error

#importing the dataset
dataset = pd.read_csv('regressionDataSet.csv')
x = dataset.iloc[:, 1:].values
y = dataset.iloc[:, 0].values

#splitting the dataset into training set and test set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 1/5)

df=pd.DataFrame()

#fitting the linear model
from sklearn.linear_model import LinearRegression
regressor1=LinearRegression()
```

```
regressor1.fit(x_train,y_train)
y_pred1=regressor1.predict(x_test)
df['pred1']=y_pred1

#fitting the polynomial model
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
poly_reg=PolynomialFeatures(degree=4)
x_poly=poly_reg.fit_transform(x_train)
regressor2=LinearRegression()
regressor2.fit(x_poly,y_train)
y_pred2=regressor2.predict(poly_reg.fit_transform(x_test))
df['pred2']=y_pred2

#fitting the elastic net model
from sklearn.linear_model import ElasticNet
regressor3=ElasticNet(max_iter=10000)
regressor3.fit(x_train,y_train)
y_pred3=regressor3.predict(x_test)
df['pred3']=y_pred3

#fitting the decision tree model
from sklearn.tree import DecisionTreeRegressor
regressor4=DecisionTreeRegressor(criterion='mae')
regressor4.fit(x_train,y_train)
y_pred4=regressor4.predict(x_test)
df['pred4']=y_pred4

#fitting the random forest model
from sklearn.ensemble import RandomForestRegressor
regressor5=RandomForestRegressor(n_estimators=500,random_state=0)
regressor5.fit(x_train,y_train)
y_pred5=regressor5.predict(x_test)
df['pred5']=y_pred5

#ensembling
y_pred=df.mean(axis=1)

#calculating r2
r2=r2_score(y_test,y_pred)

#calculating r
r=m.sqrt(r2)

#calculating error
error=mean_absolute_error(y_test,y_pred)

#calculating accuracy
accuracy = (float)(np.count_nonzero(np.array(abs(y_test - y_pred) <=
100))/np.size(y_test))*100
```

E3 - M2,M4,M6,M8,M10

```
#importing the libraries
import pandas as pd
import math as m
import numpy as np
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error

#importing the dataset
dataset = pd.read_csv('regressionDataSet.csv')
x = dataset.iloc[:, 1:].values
y = dataset.iloc[:, 0].values

#splitting the dataset into training set and test set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 1/5)

df=pd.DataFrame()

#fitting the polynomial model
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
poly_reg=PolynomialFeatures(degree=4)
x_poly=poly_reg.fit_transform(x_train)
regressor1=LinearRegression()
regressor1.fit(x_poly,y_train)
y_pred1=regressor1.predict(poly_reg.fit_transform(x_test))
df['pred1']=y_pred1

#fitting the elastic net model
from sklearn.linear_model import ElasticNet
regressor2=ElasticNet(max_iter=10000)
regressor2.fit(x_train,y_train)
y_pred2=regressor2.predict(x_test)
df['pred2']=y_pred2

#fitting the bayesian ridge model
from sklearn.linear_model import BayesianRidge
regressor3=BayesianRidge()
regressor3.fit(x_train,y_train)
y_pred3=regressor3.predict(x_test)
df['pred3']=y_pred3

#fitting the k neighbors model
from sklearn.neighbors import KNeighborsRegressor
regressor4=KNeighborsRegressor(n_neighbors=3)
regressor4.fit(x_train,y_train)
y_pred4=regressor4.predict(x_test)
df['pred4']=y_pred4

#fitting the random forest model
from sklearn.ensemble import RandomForestRegressor
regressor5=RandomForestRegressor(n_estimators=500,random_state=0)
regressor5.fit(x_train,y_train)
y_pred5=regressor5.predict(x_test)
df['pred5']=y_pred5

#ensembling
y_pred=df.mean(axis=1)

#calculating r2
```

```
r2=r2_score(y_test,y_pred)

#calculating r
r=m.sqrt(r2)

#calculating error
error=mean_absolute_error(y_test,y_pred)

#calculating accuracy
accuracy = (float)(np.count_nonzero(np.array(abs(y_test - y_pred) <=
100))/np.size(y_test))*100
```

E4 - M1,M3,M5,M7

```
#importing the libraries
import pandas as pd
import math as m
import numpy as np
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error

#importing the dataset
dataset = pd.read_csv('regressionDataSet.csv')
x = dataset.iloc[:, 1:].values
y = dataset.iloc[:, 0].values

#splitting the dataset into training set and test set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 1/5)

df=pd.DataFrame()

#fitting the linear model
from sklearn.linear_model import LinearRegression
regressor1=LinearRegression()
regressor1.fit(x_train,y_train)
y_pred1=regressor1.predict(x_test)
df['pred1']=y_pred1

#fitting the lasso model
from sklearn.linear_model import Lasso
regressor2=Lasso(max_iter=10000)
regressor2.fit(x_train,y_train)
y_pred2=regressor2.predict(x_test)
df['pred2']=y_pred2

#fitting the ridge model
from sklearn.linear_model import Ridge
regressor3=Ridge()
regressor3.fit(x_train,y_train)
y_pred3=regressor3.predict(x_test)
df['pred2']=y_pred3

#fitting the kernel ridge model
from sklearn.kernel_ridge import KernelRidge
regressor4=KernelRidge()
```

```
regressor4.fit(x_train,y_train)
y_pred4=regressor4.predict(x_test)
df['pred4']=y_pred4

#ensembling
y_pred=df.mean(axis=1)

#calculating r2
r2=r2_score(y_test,y_pred)

#calculating r
r=m.sqrt(r2)

#calculating error
error=mean_absolute_error(y_test,y_pred)

#calculating accuracy
accuracy = (float)(np.count_nonzero(np.array(abs(y_test - y_pred) <=
100))/np.size(y_test))*100
```

E5 - M1,M2,M6,M8,M10

```
#importing the libraries
import pandas as pd
import math as m
import numpy as np
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error

#importing the dataset
dataset = pd.read_csv('regressionDataSet.csv')
x = dataset.iloc[:, 1:].values
y = dataset.iloc[:, 0].values

#splitting the dataset into training set and test set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 1/5)

df=pd.DataFrame()

#fitting the linear model
from sklearn.linear_model import LinearRegression
regressor1=LinearRegression()
regressor1.fit(x_train,y_train)
y_pred1=regressor1.predict(x_test)
df['pred1']=y_pred1

#fitting the polynomial model
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
poly_reg=PolynomialFeatures(degree=4)
x_poly=poly_reg.fit_transform(x_train)
regressor2=LinearRegression()
regressor2.fit(x_poly,y_train)
y_pred2=regressor2.predict(poly_reg.fit_transform(x_test))
df['pred2']=y_pred2
```

```
#fitting the bayesian ridge model
from sklearn.linear_model import BayesianRidge
regressor3=BayesianRidge()
regressor3.fit(x_train,y_train)
y_pred3=regressor3.predict(x_test)
df['pred3']=y_pred3

#fitting the k neighbors model
from sklearn.neighbors import KNeighborsRegressor
regressor4=KNeighborsRegressor(n_neighbors=3)
regressor4.fit(x_train,y_train)
y_pred4=regressor4.predict(x_test)
df['pred4']=y_pred4

#fitting the random forest model
from sklearn.ensemble import RandomForestRegressor
regressor5=RandomForestRegressor(n_estimators=500,random_state=0)
regressor5.fit(x_train,y_train)
y_pred5=regressor5.predict(x_test)
df['pred5']=y_pred5

#ensembling
y_pred=df.mean(axis=1)

#calculating r2
r2=r2_score(y_test,y_pred)

#calculating r
r=m.sqrt(r2)

#calculating error
error=mean_absolute_error(y_test,y_pred)

#calculating accuracy
accuracy = (float)(np.count_nonzero(np.array(abs(y_test - y_pred) <=
100))/np.size(y_test))*100
```

Q1.3 Study 5 feature selection techniques for given regression data set and report Top five features.

Table 1.3: Study of feature selection techniques

Feature Selection Technique	Top 5 Features
T1 - SelectKBest	F5,F6,F8,F9,F10
T2 - SelectFdr	F1,F2,F4,F5,F6
T3 - RFE	F1,F2,F3,F4,F14
T4 - SelectFromModel	F1,F2,F3,F4,F14
T5 - SelectFwe	F1,F2,F3,F4,F5

Feature Selection

```
#importing the libraries
```

```
import pandas as pd

#importing the dataset
dataset = pd.read_csv('regressionDataSet.csv')
x = dataset.iloc[:, 1:].values
y = dataset.iloc[:, 0].values

#feature selector 1
from sklearn.feature_selection import SelectKBest
fs1=SelectKBest(k=5)
x_new1=fs1.fit_transform(x,y)

#feature selector 2
from sklearn.feature_selection import SelectFdr
fs2=SelectFdr()
x_new2=fs2.fit_transform(x,y)

#feature selector 3
from sklearn.linear_model import LinearRegression
estimator = LinearRegression()
from sklearn.feature_selection import RFE
fs3=RFE(estimator,5)
x_new3=fs3.fit_transform(x,y)

#feature selector 4
from sklearn.feature_selection import SelectFromModel
fs4=SelectFromModel(estimator)
x_new4=fs4.fit_transform(x,y)

#feature selector 5
from sklearn.feature_selection import SelectFwe
fs5=SelectFwe()
x_new5=fs5.fit_transform(x,y)
```

PART II

[For Classification Dataset | filename: classificationDataSet.csv]

Q2.1 Compare the performance of 10 machine learning models for given classification data set for the data partition of 70-30%.

Table 2.1: Comparative Performance Study of Machine Learning Models

Model	Method	Package	Sensitivity	Specificity	Precision	Recall	Accuracy

M1	Logistic Regression	sklearn.linear_model	0.6744	0.3770	0.5357	0.6744	0.5304
M2	SGD	sklearn.linear_model	0.4262	0.5175	0.4840	0.4262	0.4704
M3	Naive Bayes	sklearn.naive_bayes	0.5560	0.4565	0.5227	0.5560	0.5080
M4	SVC	sklearn.svm	0.6298	0.4070	0.5444	0.6298	0.5250
M5	Gaussian Process	sklearn.gaussian_process	0.5942	0.4265	0.5154	0.5942	0.5114
M6	K-Nieghbors	sklearn.neighbors	0.5833	0.4299	0.5277	0.5833	0.5100
M7	Nearest Centroid	sklearn.neighbors	0.5670	0.5015	0.5426	0.5670	0.5350
M8	Radius Nieghbors	sklearn.neighbors	1.0	0.0	0.502	1.0	0.502
M9	Decision Tree	sklearn.tree	0.5194	0.4984	0.5352	0.5194	0.5094
M10	Random Forest	sklearn.ensemble	0.5958	0.41658	0.49279	0.59589	0.504

M1 - Logisctic Regression Model

```
#importing the libraries
import pandas as pd
from sklearn.metrics import confusion_matrix

#importing the dataset
dataset = pd.read_csv('classificationDataSet.csv')
x = dataset.iloc[:,1:].values
y = dataset.iloc[:, 0].values

#splitting the dataset into training set and test set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 1/5)

#feature scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)

#fitting the model on the training test
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(max_iter=10000)
classifier.fit(x_train, y_train)

#predicting the test set results
y_pred = classifier.predict(x_test)

#confusion matrix
cm=confusion_matrix(y_test,y_pred)
tn=cm[0][0]
fp=cm[0][1]
fn=cm[1][0]
tp=cm[1][1]

#sensitiviry
```

```
sensitivity=(tp)/(tp+fn)

#specificity
specificity=(tn)/(tn+fp)

#precision
precision=(tp)/(tp+fp)

#recall
recall=(tp)/(tp+fn)

#accuracy
accuracy=(tn+tp)/(tp+fn+fp+tn)
```

M2 - SGD Model

```
#importing the libraries
import pandas as pd
from sklearn.metrics import confusion_matrix

#importing the dataset
dataset = pd.read_csv('classificationDataSet.csv')
x = dataset.iloc[:,1:].values
y = dataset.iloc[:, 0].values

#splitting the dataset into training set and test set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 1/5)

#feature scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)

#fitting the model on the training test
from sklearn.linear_model import SGDClassifier
classifier = SGDClassifier()
classifier.fit(x_train, y_train)

#predicting the test set results
y_pred = classifier.predict(x_test)

#confusion matrix
cm=confusion_matrix(y_test,y_pred)
tn=cm[0][0]
fp=cm[0][1]
fn=cm[1][0]
tp=cm[1][1]

#sensitiviry
sensitivity=(tp)/(tp+fn)

#specificity
specificity=(tn)/(tn+fp)
```

```
#precision
precision=(tp)/(tp+fp)

#recall
recall=(tp)/(tp+fn)

#accuracy
accuracy=(tn+tp)/(tp+fn+fp+tn)
```

M3 - Naive Bayes Model

```
#importing the libraries
import pandas as pd
from sklearn.metrics import confusion_matrix

#importing the dataset
dataset = pd.read_csv('classificationDataSet.csv')
x = dataset.iloc[:,1:].values
y = dataset.iloc[:, 0].values

#splitting the dataset into training set and test set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 1/5)

#feature scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)

#fitting the model on the training test
from sklearn.naive_bayes import GaussianNB
classifier=GaussianNB()
classifier.fit(x_train, y_train)

#predicting the test set results
y_pred = classifier.predict(x_test)

#confusion matrix
cm=confusion_matrix(y_test,y_pred)
tn=cm[0][0]
fp=cm[0][1]
fn=cm[1][0]
tp=cm[1][1]

#sensitiviry
sensitivity=(tp)/(tp+fn)

#specificity
specificity=(tn)/(tn+fp)

#precision
precision=(tp)/(tp+fp)

#recall
recall=(tp)/(tp+fn)
```

```
#accuracy
accuracy=(tn+tp)/(tp+fn+fp+tn)
```

M4 - SVC Model

```
#importing the libraries
import pandas as pd
from sklearn.metrics import confusion_matrix

#importing the dataset
dataset = pd.read_csv('classificationDataSet.csv')
x = dataset.iloc[:,1:].values
y = dataset.iloc[:, 0].values

#splitting the dataset into training set and test set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 1/5)

#feature scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)

#fitting the model on the training test
from sklearn.svm import SVC
classifier=SVC(max_iter=10000,kernel='rbf')
classifier.fit(x_train, y_train)

#predicting the test set results
y_pred = classifier.predict(x_test)

#confusion matrix
cm=confusion_matrix(y_test,y_pred)
tn=cm[0][0]
fp=cm[0][1]
fn=cm[1][0]
tp=cm[1][1]

#sensitiviry
sensitivity=(tp)/(tp+fn)

#specificity
specificity=(tn)/(tn+fp)

#precision
precision=(tp)/(tp+fp)

#recall
recall=(tp)/(tp+fn)

#accuracy
accuracy=(tn+tp)/(tp+fn+fp+tn)
```

M5 - Gaussian Process Model

```
#importing the libraries
import pandas as pd
from sklearn.metrics import confusion_matrix

#importing the dataset
dataset = pd.read_csv('classificationDataSet.csv')
x = dataset.iloc[:,1:].values
y = dataset.iloc[:, 0].values

#splitting the dataset into training set and test set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 1/5)

#feature scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)

#fitting the model on the training test
from sklearn.gaussian_process import GaussianProcessClassifier
classifier = GaussianProcessClassifier()
classifier.fit(x_train, y_train)

#predicting the test set results
y_pred = classifier.predict(x_test)

#confusion matrix
cm=confusion_matrix(y_test,y_pred)
tn=cm[0][0]
fp=cm[0][1]
fn=cm[1][0]
tp=cm[1][1]

#sensitiviry
sensitivity=(tp)/(tp+fn)

#specificity
specificity=(tn)/(tn+fp)

#precision
precision=(tp)/(tp+fp)

#recall
recall=(tp)/(tp+fn)

#accuracy
accuracy=(tn+tp)/(tp+fn+fp+tn)
```

M6 - K-Neighbors Model

```
#importing the libraries
import pandas as pd
```

```
from sklearn.metrics import confusion_matrix

#importing the dataset
dataset = pd.read_csv('classificationDataSet.csv')
x = dataset.iloc[:,1:].values
y = dataset.iloc[:, 0].values

#splitting the dataset into training set and test set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 1/5)

#feature scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)

#fitting the model on the training test
from sklearn.neighbors import KNeighborsClassifier
classifier=KNeighborsClassifier(n_neighbors=100)
classifier.fit(x_train, y_train)

#predicting the test set results
y_pred = classifier.predict(x_test)

#confusion matrix
cm=confusion_matrix(y_test,y_pred)
tn=cm[0][0]
fp=cm[0][1]
fn=cm[1][0]
tp=cm[1][1]

#sensitiviry
sensitivity=(tp)/(tp+fn)

#specificity
specificity=(tn)/(tn+fp)

#precision
precision=(tp)/(tp+fp)

#recall
recall=(tp)/(tp+fn)

#accuracy
accuracy=(tn+tp)/(tp+fn+fp+tn)
```

M7 - Nearest Centroid Model

```
#importing the libraries
import pandas as pd
from sklearn.metrics import confusion_matrix

#importing the dataset
dataset = pd.read_csv('classificationDataSet.csv')
x = dataset.iloc[:,1:].values
```

```
y = dataset.iloc[:, 0].values

#splitting the dataset into training set and test set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 1/5)

#feature scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)

#fitting the model on the training test
from sklearn.neighbors import NearestCentroid
classifier = NearestCentroid()
classifier.fit(x_train, y_train)

#predicting the test set results
y_pred = classifier.predict(x_test)

#confusion matrix
cm=confusion_matrix(y_test,y_pred)
tn=cm[0][0]
fp=cm[0][1]
fn=cm[1][0]
tp=cm[1][1]

#sensitiviry
sensitivity=(tp)/(tp+fn)

#specificity
specificity=(tn)/(tn+fp)

#precision
precision=(tp)/(tp+fp)

#recall
recall=(tp)/(tp+fn)

#accuracy
accuracy=(tn+tp)/(tp+fn+fp+tn)
```

M8 - Radius Neighbors Model

```
#importing the libraries
import pandas as pd
from sklearn.metrics import confusion_matrix

#importing the dataset
dataset = pd.read_csv('classificationDataSet.csv')
x = dataset.iloc[:,1:].values
y = dataset.iloc[:, 0].values

#splitting the dataset into training set and test set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 1/5)
```

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

#fitting the model on the training test
from sklearn.neighbors import RadiusNeighborsClassifier
classifier= RadiusNeighborsClassifier(radius=10)
classifier.fit(x_train, y_train)

#predicting the test set results
y_pred = classifier.predict(x_test)

#confusion matrix
cm=confusion_matrix(y_test,y_pred)
tn=cm[0][0]
fp=cm[0][1]
fn=cm[1][0]
tp=cm[1][1]

#sensitiviry
sensitivity=(tp)/(tp+fn)

#specificity
specificity=(tn)/(tn+fp)

#precision
precision=(tp)/(tp+fp)

#recall
recall=(tp)/(tp+fn)

#accuracy
accuracy=(tn+tp)/(tp+fn+fp+tn)
```

M9 - Decision Tree Model

```
#importing the libraries
import pandas as pd
from sklearn.metrics import confusion_matrix

#importing the dataset
dataset = pd.read_csv('classificationDataSet.csv')
x = dataset.iloc[:,1:].values
y = dataset.iloc[:, 0].values

#splitting the dataset into training set and test set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 1/5)

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

```
#fitting the model on the training test
from sklearn.tree import DecisionTreeClassifier
classifier=DecisionTreeClassifier(criterion='entropy')
classifier.fit(x_train, y_train)

#predicting the test set results
y_pred = classifier.predict(x_test)

#confusion matrix
cm=confusion_matrix(y_test,y_pred)
tn=cm[0][0]
fp=cm[0][1]
fn=cm[1][0]
tp=cm[1][1]

#sensitiviry
sensitivity=(tp)/(tp+fn)

#specificity
specificity=(tn)/(tn+fp)

#precision
precision=(tp)/(tp+fp)

#recall
recall=(tp)/(tp+fn)

#accuracy
accuracy=(tn+tp)/(tp+fn+fp+tn)
```

M10 - Random Forest Model

```
#importing the libraries
import pandas as pd
from sklearn.metrics import confusion_matrix

#importing the dataset
dataset = pd.read_csv('classificationDataSet.csv')
x = dataset.iloc[:,1:].values
y = dataset.iloc[:, 0].values

#splitting the dataset into training set and test set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 1/5)

#feature scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)

#fitting the model on the training test
from sklearn.ensemble import RandomForestClassifier
classifier=RandomForestClassifier(n_estimators=500,criterion='entropy')
classifier.fit(x_train, y_train)
```

```
#predicting the test set results
y_pred = classifier.predict(x_test)

#confusion matrix
cm=confusion_matrix(y_test,y_pred)
tn=cm[0][0]
fp=cm[0][1]
fn=cm[1][0]
tp=cm[1][1]

#sensitiviry
sensitivity=(tp)/(tp+fn)

#specificity
specificity=(tn)/(tn+fp)

#precision
precision=(tp)/(tp+fp)

#recall
recall=(tp)/(tp+fn)

#accuracy
accuracy=(tn+tp)/(tp+fn+fp+tn)
```

Q2.2 Ensemble the models from Table 2.1 for given classification data set on data partition of 70-30%.

Table 2.2: Result analysis of ensemble models

Model	Combination	Sensitivity	Specificity	Precision	Recall	Accuracy
E1	M1,M5,M6,M7,M10	0.7401	0.2977	0.5211	0.7401	0.5224
E2	M1,M2,M4	0.5856	0.4603	0.5284	0.5856	0.5240
E3	M2,M4,M6,M8,M10	0.5472	0.4705	0.51624	0.54724	0.50949
E4	M5,M7,M8	0.7706	0.2723	0.5223	0.7706	0.5254
E5	M1,M2,M6,M8,M10	0.5659	0.4502	0.5152	0.5659	0.5090

Ensembling

```
#importing the libraries
import pandas as pd
from sklearn.metrics import confusion_matrix

#importing the dataset
dataset = pd.read_csv('classificationDataSet.csv')
x = dataset.iloc[:,1:].values
y = dataset.iloc[:, 0].values

#splitting the dataset into training set and test set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 1/5)
```

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

df=pd.DataFrame()

#fitting the logistic regression model on the training test
from sklearn.linear_model import LogisticRegression
classifier1 = LogisticRegression(max_iter=10000)
classifier1.fit(x_train, y_train)
y_pred1 = classifier1.predict(x_test)
df['pred1']=y_pred1

#fitting the model on the training test
from sklearn.linear_model import SGDClassifier
classifier2 = SGDClassifier()
classifier2.fit(x_train, y_train)
y_pred2 = classifier2.predict(x_test)
df['pred2']=y_pred2

#fitting the gaussian process model on the training set
from sklearn.gaussian_process import GaussianProcessClassifier
classifier3 = GaussianProcessClassifier()
classifier3.fit(x_train, y_train)
y_pred3 = classifier3.predict(x_test)
df['pred3']=y_pred3

#fitting the model on the training test
from sklearn.naive_bayes import GaussianNB
classifier4=GaussianNB()
classifier4.fit(x_train, y_train)
y_pred4 = classifier4.predict(x_test)
df['pred4']=y_pred4

#fitting the model on the training test
from sklearn.gaussian_process import GaussianProcessClassifier
classifier5 = GaussianProcessClassifier()
classifier5.fit(x_train, y_train)
y_pred5 = classifier5.predict(x_test)
df['pred5']=y_pred5

#fitting the model on the training test
from sklearn.neighbors import KNeighborsClassifier
classifier6=KNeighborsClassifier(n_neighbors=100)
classifier6.fit(x_train, y_train)
y_pred6 = classifier6.predict(x_test)
df['pred6']=y_pred6

#fitting the model on the training test
from sklearn.neighbors import RadiusNeighborsClassifier
classifier7= RadiusNeighborsClassifier(radius=10)
classifier7.fit(x_train, y_train)
y_pred7 = classifier7.predict(x_test)
df['pred7']=y_pred7
```

```
#fitting the model on the training test
from sklearn.neighbors import NearestCentroid
classifier8 = NearestCentroid()
classifier8.fit(x_train, y_train)
y_pred8 = classifier8.predict(x_test)
df['pred8']=y_pred8

#fitting the model on the training test
from sklearn.tree import DecisionTreeClassifier
classifier9=DecisionTreeClassifier(criterion='entropy')
classifier9.fit(x_train, y_train)
y_pred9 = classifier9.predict(x_test)
df['pred9']=y_pred9

#fitting the model on the training test
from sklearn.ensemble import RandomForestClassifier
classifier10=RandomForestClassifier(n_estimators=500,criterion='entropy')
classifier10.fit(x_train, y_train)
y_pred10 = classifier10.predict(x_test)
df['pred10']=y_pred10

#ensembling
df1 = pd.DataFrame()
df2 = pd.DataFrame()
df3 = pd.DataFrame()
df4 = pd.DataFrame()
df5 = pd.DataFrame()

df1 = df[['pred1','pred5','pred6','pred7','pred10']].copy()
df2 = df[['pred1','pred2','pred4']].copy()
df3 = df[['pred2','pred4','pred6','pred8','pred10']].copy()
df4 = df[['pred5','pred7','pred8']].copy()
df5 = df[['pred1','pred2','pred6','pred8','pred10']].copy()

test=df1
comp=len(test.columns)
y_pred=(test==1).astype(int).sum(axis=1)/comp > 0.5
y_pred=y_pred.astype(int)

#confusion matrix
cm=confusion_matrix(y_test,y_pred)
tn=cm[0][0]
fp=cm[0][1]
fn=cm[1][0]
tp=cm[1][1]

#sensitiviry
sensitivity=(tp)/(tp+fn)

#specificity
specificity=(tn)/(tn+fp)

#precision
precision=(tp)/(tp+fp)

#recall
recall=(tp)/(tp+fn)
```

```
#accuracy
accuracy=(tn+tp)/(tp+fn+fp+tn)
```

Q2.3 Study 5 feature selection techniques for given classification data set and report Top five features.

Table 2.3: Study of feature selection techniques

Feature Selection Technique	Top 5 Features
T1 - SelectKBest	F14,F15,F16,F17,F20
T2 - SelectFdr	F4,F5,F13,F14,F15
T3 - RFE	F3,F4,F14,F17,F18
T4 - SelectFromModel	F1,F3,F4,F9,F11
T5 - SelectFwe	F5,F14,F15,F16,F17

Feature Selection

```
#importing the libraries
import pandas as pd

#importing the dataset
dataset = pd.read_csv('classificationDataSet.csv')
x = dataset.iloc[:, 1:].values
y = dataset.iloc[:, 0].values

#feature selector 1
from sklearn.feature_selection import SelectKBest
fs1=SelectKBest(k=5)
x_new1=fs1.fit_transform(x,y)

#feature selector 2
from sklearn.feature_selection import SelectFdr
fs2=SelectFdr()
x_new2=fs2.fit_transform(x,y)

#feature selector 3
from sklearn.linear_model import LogisticRegression
estimator = LogisticRegression()
from sklearn.feature_selection import RFE
fs3=RFE(estimator,5)
x_new3=fs3.fit_transform(x,y)

#feature selector 4
from sklearn.feature_selection import SelectFromModel
fs4=SelectFromModel(estimator)
x_new4=fs4.fit_transform(x,y)

#feature selector 5
from sklearn.feature_selection import SelectFwe
fs5=SelectFwe()
x_new5=fs5.fit_transform(x,y)
```