ISTS WOMEN'S ENGINEERING COLLEGE

(Affiliated To JNTUK, Kakinada)

ISTS

INTERNATIONAL SCHOOL OF TECHNOLOGY
AND SCIENCES FOR WOMEN

IS0:9001:2015

4 DAYS BOOTCAMP AND 24 HOURS HACKATHON ON DATA SCIENCE WITH PYTHON

सत्यमेव जयते
Ministry of Education
Government of India

BRAIN O VISION
Brain O Vision Solutions India Pvt. Ltd...

AICTE

Team members:

1. A V S Leelavathi
2. V Ramanamma
3. K Monika
4. P Puja
5. G Yamini
6. K Navya

Project uploading levels:

Level 1 (Problem Statement)
Level 2 (Feature Engineering)
Level 3 Correction
Level 4 (Evaluation Metrics) & Conclusion

⇨ PROBLEM STATEMENT

What is a problem statement ?

A problem statement is a description of an identified issue that a company or business has, detailing methods needed to address and overcome it. It is a problem solving **identifies the gap between the current problem and goal**.

How to write a problem statement ?

We can problem statement in five steps

1. Identify the issue
2. Begin with your ideal situation
3. Describe current gaps
4. State the consequent of the problem
5. Propose addressing the issue

INDUSTRY : AUTOMOBILES

DEPARTMENT : MANUFACTURING

PROBLEM : SALES SLUMP

CAUSE : CHANGES IN CUSTOMER EXPECTATIONS

SOLUTION : ADAPTING TECHNOLOGIES

PROBLEM STATEMENT:

Predicting the sales slump in automobile industry while varies in customer expectations we can predict this with the help of adapting technologies.

Automobile industry:

It is a broad range of companies and organizations involved in the design, development, manufacturing, marketing, and selling of motor vehicles. It is one largest economic sectors in terms of revenue. Software and Hardware engineers in the automotive sector work for automotive manufacturers.

Example : Sonalika Tractors

Sonalika Tractors faced a drop of 14.41% in retail tractor sales in India 2023. The company sold 8,293 units in November 2023 compared to 9,689 units in November 2022.

TAFE Limited gained its market share by 0.09%. The company recorded a drop in retail tractor sales by 20.65%.

References : Referred from google websites.

1.

https://getjerry.com/car-repair/common-problems-with-honda

2.

https://www.google.com/search?q=which+company+is+sales+down+in+tractor+industry&rlz=1C1CHBF_enIN1081IN1082&oq=which+company+is+sales+down+in+tactors&gs_lcrp=EgZjaHJvbWUqCQgBECEYChigATIGCAAQRRg5MgkIARAhGAoYoAEyCQgCECEYChigATIJCAMQIRgKGKABMgYIBBAhGAoyBwgFECEYnwUyBwgGECEYnwUyBwgHECEYnwUyBwgIECEYnwUyBwgJECEYnwXSAQkxOTU5NWowajeoAgCwAgA&sourceid=chrome&ie=UTF-

⇨ FEATURES IN AUTOMOBILES INDUSTRY:

1. Development--Development------------------------------
2. Manufacturing--Manufacturing--------
3. Marketing
4. Selling--x-----------------------
5. Organization involved in the design
6. Repairing
7. Quality--satisfaction rate---------------------
8. Modification of motor vehicles
9. Innovation
10. Reliability--x-----------------------------
11. Performance
12. Cost____Benifits_________________
13. Interest Rate
14. Economic Growth
15. Safety and Aesthetics____Benifits______________
16. Exchange Rate

17. Employment Rate
18. Car Sharing Apps+
19. Products and technology
20. Vehicle type---Brand------------------------------------
21. Branding and marketing---satisfaction rate------------------------
22. Globalized \production
23. Comfort--Benefit---------
24. Predictive maintenance____Benifits_ ____________________
25. Autonomous driving
26. Technology--satisfaction rate-------------------------------

Performance-------------------*******

Quality------------------*******

Rating------------------********

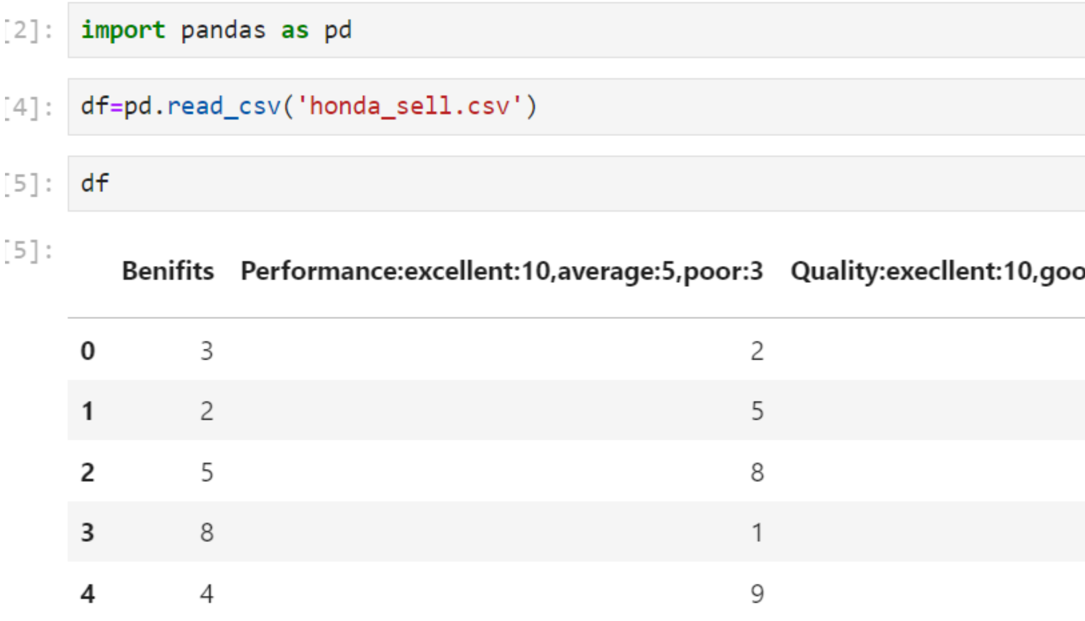
Satisfaction------------*****

Benefits------------*******

Sonalika Tractors Dataset

1).Import pandas as pd

You have imported the pandas library in Python With pandas, you can easily work with structured data, perform data manipulation, analysis, and visualization tasks. Not only pandas libray we can take any library for importing data.

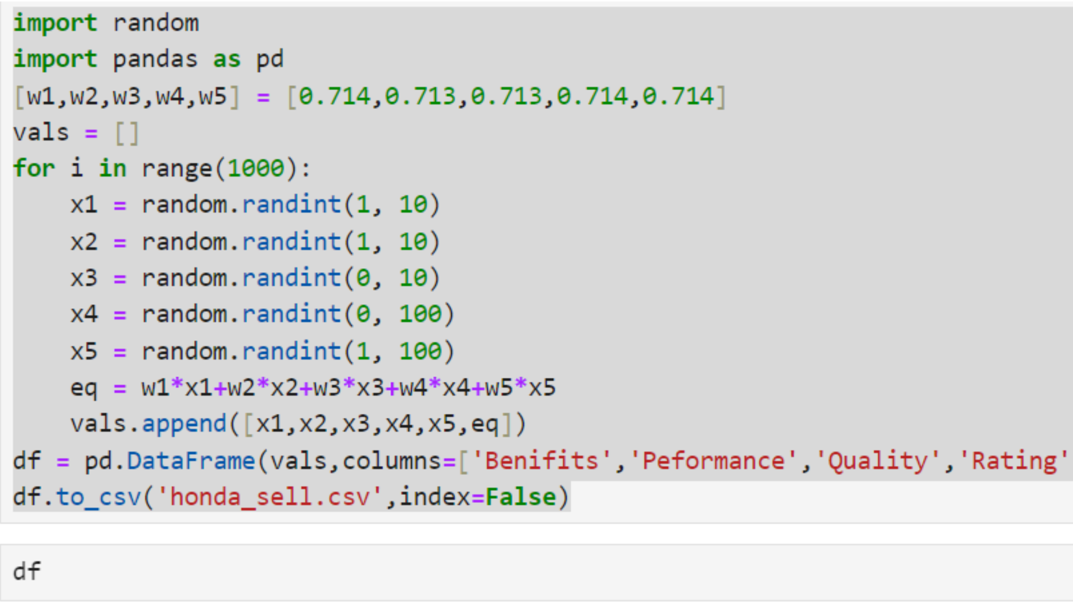


```
[2]: import pandas as pd

[4]: df=pd.read_csv('honda_sell.csv')

[5]: df

[5]:
```

	Benifits	Performance:excellent:10,average:5,poor:3	Quality:execllent:10,goo
0	3	2	
1	2	5	
2	5	8	
3	8	1	
4	4	9	

In this above code we can use pandas library to import the data set. Here we can take df as variable of the data set.

1) To generate data to a specified range.

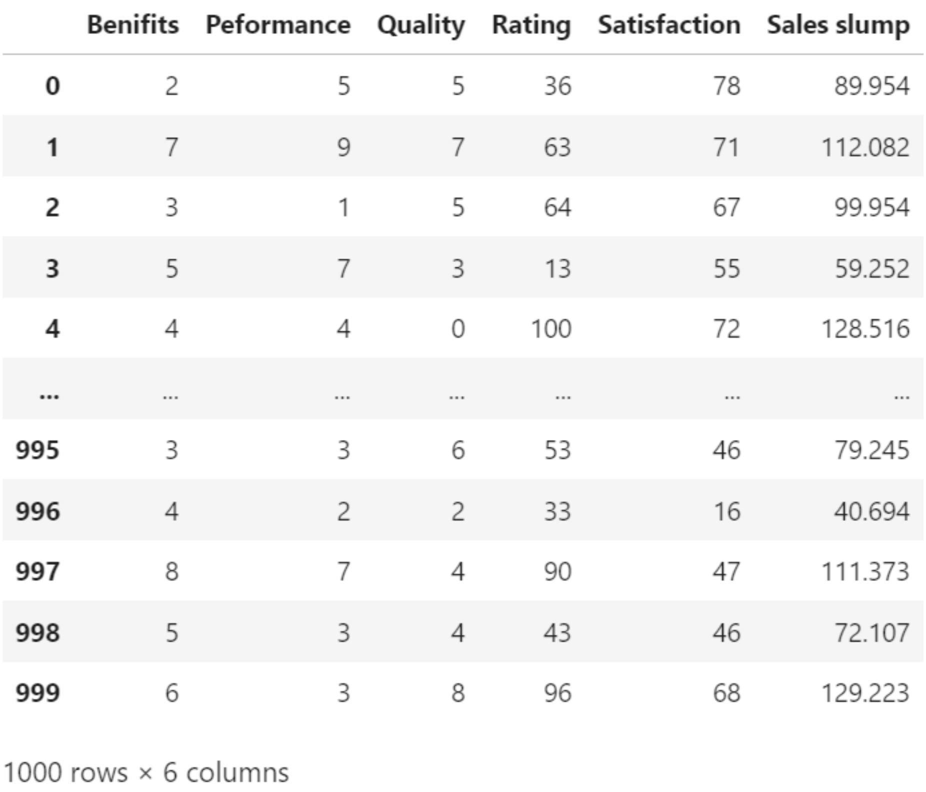


```
import random
import pandas as pd
[w1,w2,w3,w4,w5] = [0.714,0.713,0.713,0.714,0.714]
vals = []
for i in range(1000):
    x1 = random.randint(1, 10)
    x2 = random.randint(1, 10)
    x3 = random.randint(0, 10)
    x4 = random.randint(0, 100)
    x5 = random.randint(1, 100)
    eq = w1*x1+w2*x2+w3*x3+w4*x4+w5*x5
    vals.append([x1,x2,x3,x4,x5,eq])
df = pd.DataFrame(vals,columns=['Benifits','Peformance','Quality','Rating'
df.to_csv('honda_sell.csv',index=False)
```

```
df
```

Random.randint() is a function from the Random module in Python that generates a random integer within a specified range

2) .Output for the above algorithm.

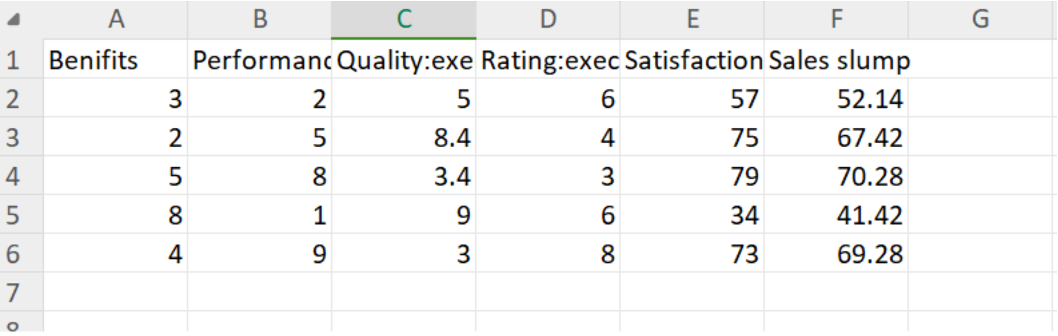


	Benifits	Peformance	Quality	Rating	Satisfaction	Sales slump
0	2	5	5	36	78	89.954
1	7	9	7	63	71	112.082
2	3	1	5	64	67	99.954
3	5	7	3	13	55	59.252
4	4	4	0	100	72	128.516
...	...	...	...	...	...	...
995	3	3	6	53	46	79.245
996	4	2	2	33	16	40.694
997	8	7	4	90	47	111.373
998	5	3	4	43	46	72.107
999	6	3	8	96	68	129.223

1000 rows × 6 columns

Here we can generate the dataset upto 1000 times.

3).Dataset



	A	B	C	D	E	F	G
1	Benifits	Performanc	Quality:exe	Rating:exec	Satisfaction	Sales slump	
2	3	2	5	6	57	52.14	
3	2	5	8.4	4	75	67.42	
4	5	8	3.4	3	79	70.28	
5	8	1	9	6	34	41.42	
6	4	9	3	8	73	69.28	
7							

ALGORITHMS

Algorithm:

A set of finite rules or instructions to be followed in calculations or other problem-solving operations.In these algorithms we have some types which are…

REGRESSION:

Regression is a statistical method used to model the relationship between a dependent variable and one or more independent variables.In regression we have two types .

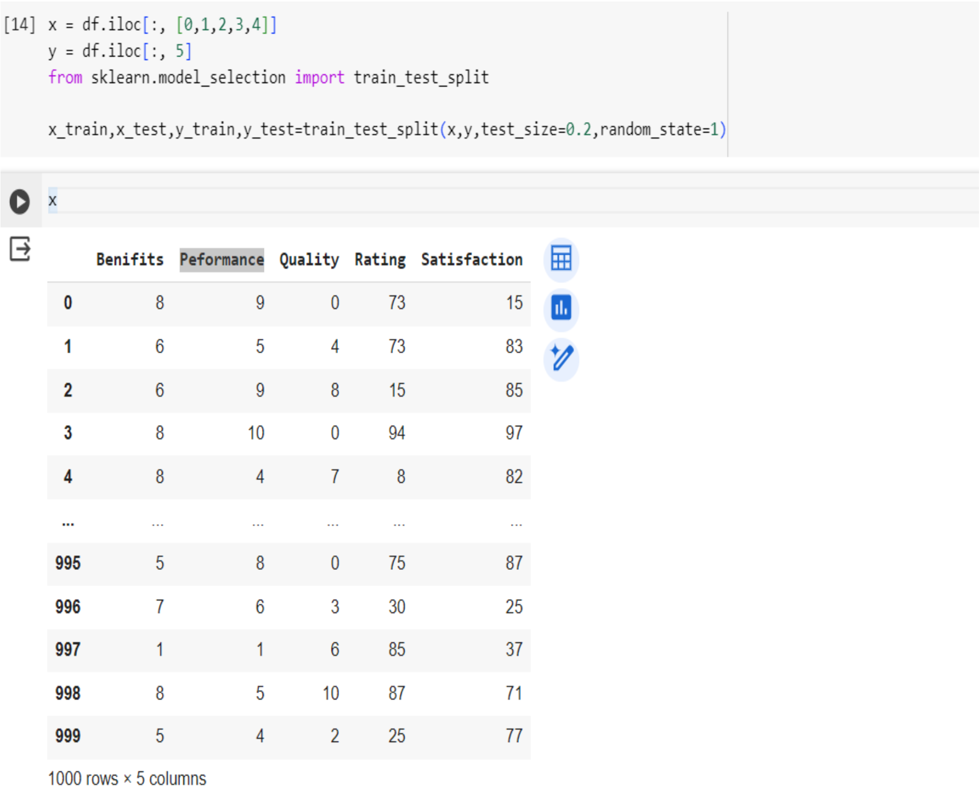
Linear regression :-

Linear regression is a basic and straightforward machine learning model used for regression analysis. It involves finding the relationship between two variables, i.e., the dependent variable and the independent variable. This model is used to predict a continuous outcome variable based on one or more predictor variables

MultipleLinear Regression:-

Multiple linear regression attempts to model the relationship between two or more independent variables and a dependent variable by fitting a linear equation to observed data. Every value of the independent variable x is associated with a value of the dependent variable

1).Asign values variables.



```
[14] x = df.iloc[:, [0,1,2,3,4]]
     y = df.iloc[:, 5]
     from sklearn.model_selection import train_test_split

     x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=1)
```

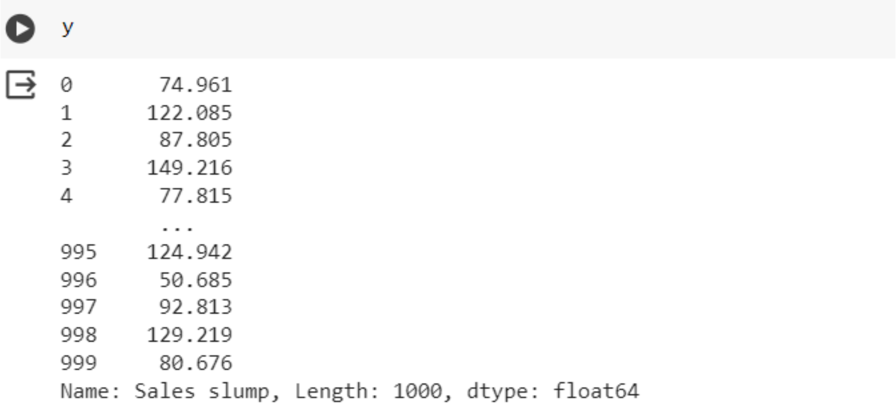
```
x
```

	Benifits	Peformance	Quality	Rating	Satisfaction
0	8	9	0	73	15
1	6	5	4	73	83
2	6	9	8	15	85
3	8	10	0	94	97
4	8	4	7	8	82
...	...	...	...	...	...
995	5	8	0	75	87
996	7	6	3	30	25
997	1	1	6	85	37
998	8	5	10	87	71
999	5	4	2	25	77

1000 rows × 5 columns

Here we can take two variables x and y ,then we print values of x.

2).Printing a value.

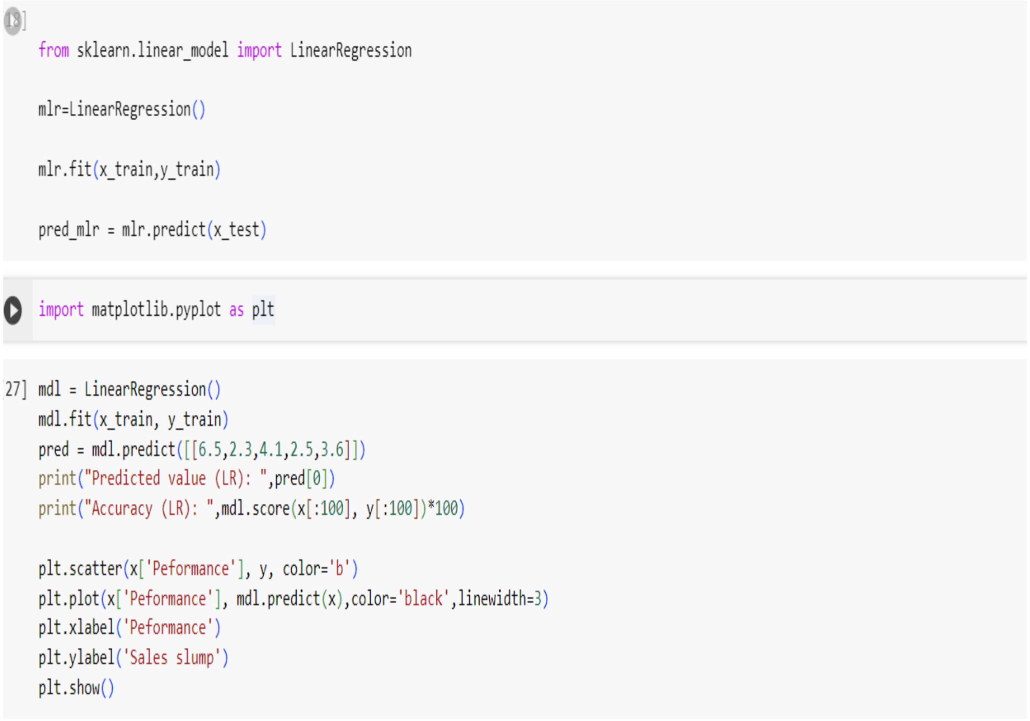


```
y
```

```
0       74.961
1      122.085
2       87.805
3      149.216
4       77.815
        ...
995    124.942
996     50.685
997     92.813
998    129.219
999     80.676
Name: Sales slump, Length: 1000, dtype: float64
```

After that we can the values of y.

3).Linear regression by importing matplotlib.



```
from sklearn.linear_model import LinearRegression

mlr=LinearRegression()

mlr.fit(x_train,y_train)

pred_mlr = mlr.predict(x_test)
```

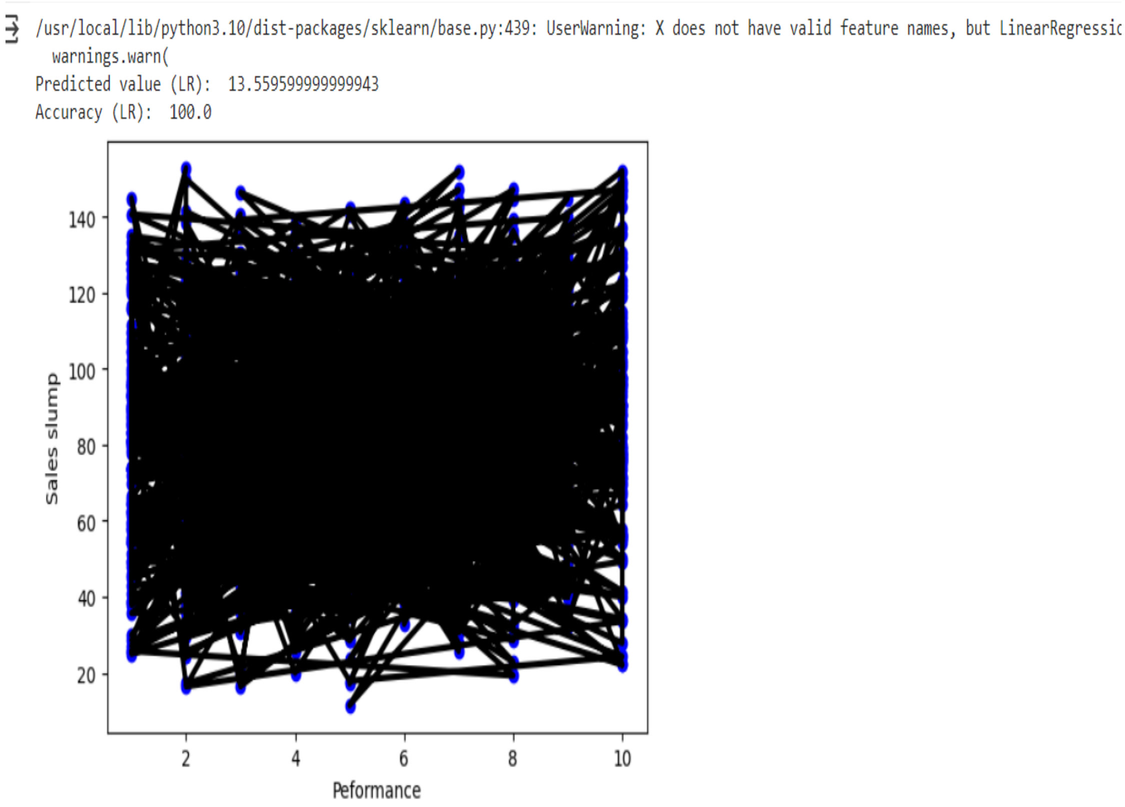
```
import matplotlib.pyplot as plt
```

```
mdl = LinearRegression()
mdl.fit(x_train, y_train)
pred = mdl.predict([[6.5,2.3,4.1,2.5,3.6]])
print("Predicted value (LR): ",pred[0])
print("Accuracy (LR): ",mdl.score(x[:100], y[:100])*100)

plt.scatter(x['Peformance'], y, color='b')
plt.plot(x['Peformance'], mdl.predict(x),color='black',linewidth=3)
plt.xlabel('Peformance')
plt.ylabel('Sales slump')
plt.show()
```

At the beginning we can import ‘matplotlib’ library as plt after that we use a

Scatter function to the attributes of x and y.Then we give linewidth, colour to that axis.

Output:



```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegressic
  warnings.warn(
Predicted value (LR):  13.559599999999943
Accuracy (LR):  100.0
```

It is the output of the above algorithm.

Decision Trees:-

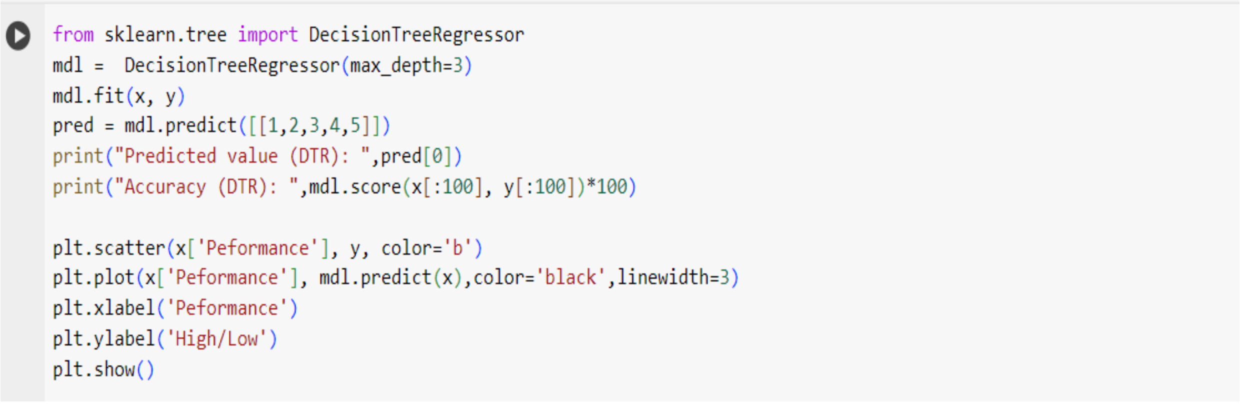
A decision tree is a simple and intuitive model that is used for both

regression and classification problems. It is a tree-like structure where each node

represents a feature or attribute, and each branch represents a decision rule.

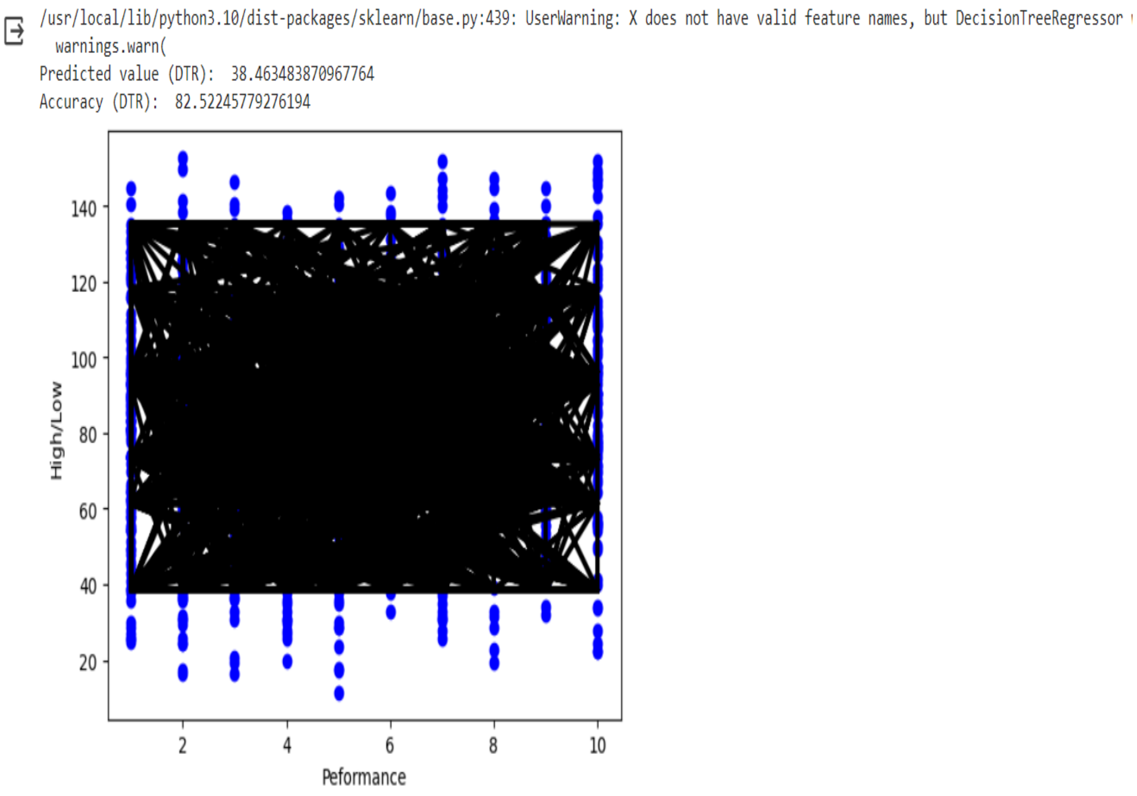
Decision trees are particularly useful in situations where the data has multiple

variables and is non-linear



```
from sklearn.tree import DecisionTreeRegressor
mdl =  DecisionTreeRegressor(max_depth=3)
mdl.fit(x, y)
pred = mdl.predict([[1,2,3,4,5]])
print("Predicted value (DTR): ",pred[0])
print("Accuracy (DTR): ",mdl.score(x[:100], y[:100])*100)

plt.scatter(x['Peformance'], y, color='b')
plt.plot(x['Peformance'], mdl.predict(x),color='black',linewidth=3)
plt.xlabel('Peformance')
plt.ylabel('High/Low')
plt.show()
```

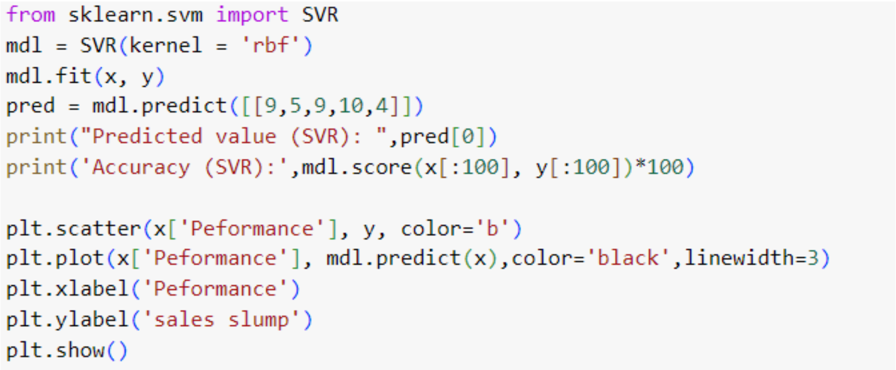
Output:



```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but DecisionTreeRegressor
  warnings.warn(
Predicted value (DTR):  38.463483870967764
Accuracy (DTR):  82.52245779276194
```

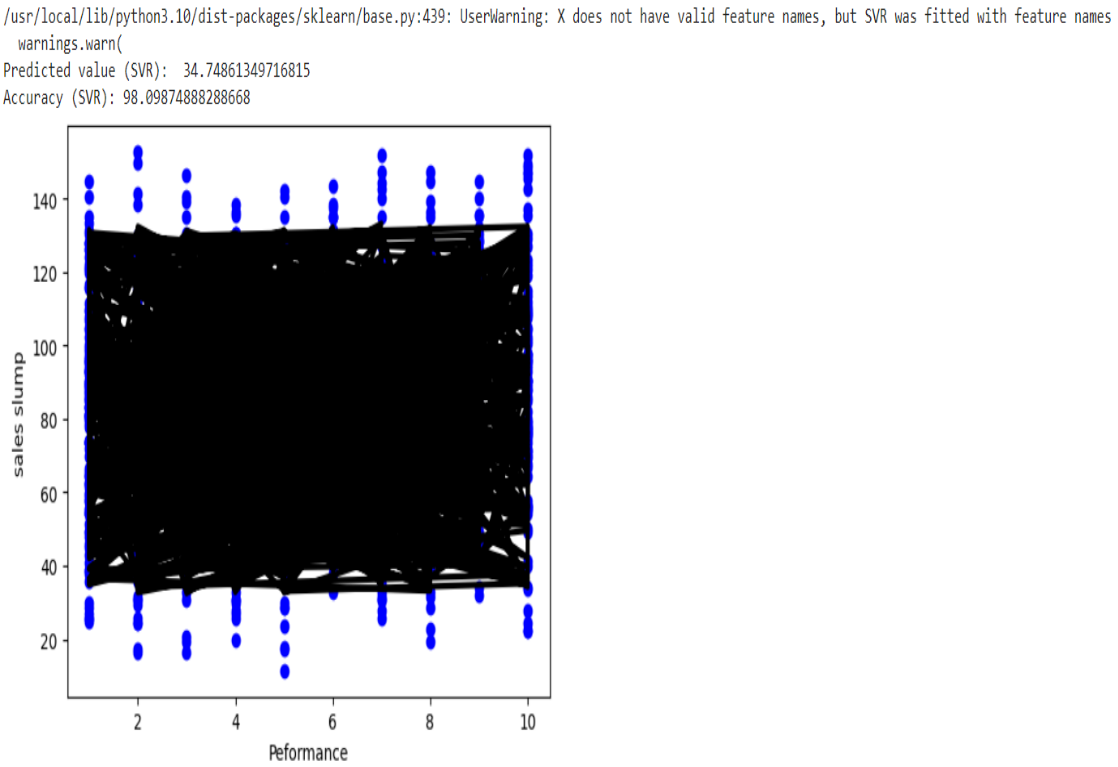
Support vector machines (SVM) :-

Support vector machines (SVM) are a supervised learning algorithm used for classification and regression analysis. SVM tries to find the best hyperplane that separates the data points into different classes, with maximum margin. SVM can also handle non-linearly separable data by transforming the data into a higher-dimensional space.



```
from sklearn.svm import SVR
mdl = SVR(kernel = 'rbf')
mdl.fit(x, y)
pred = mdl.predict([[9,5,9,10,4]])
print("Predicted value (SVR): ",pred[0])
print('Accuracy (SVR):',mdl.score(x[:100], y[:100])*100)

plt.scatter(x['Peformance'], y, color='b')
plt.plot(x['Peformance'], mdl.predict(x),color='black',linewidth=3)
plt.xlabel('Peformance')
plt.ylabel('sales slump')
plt.show()
```

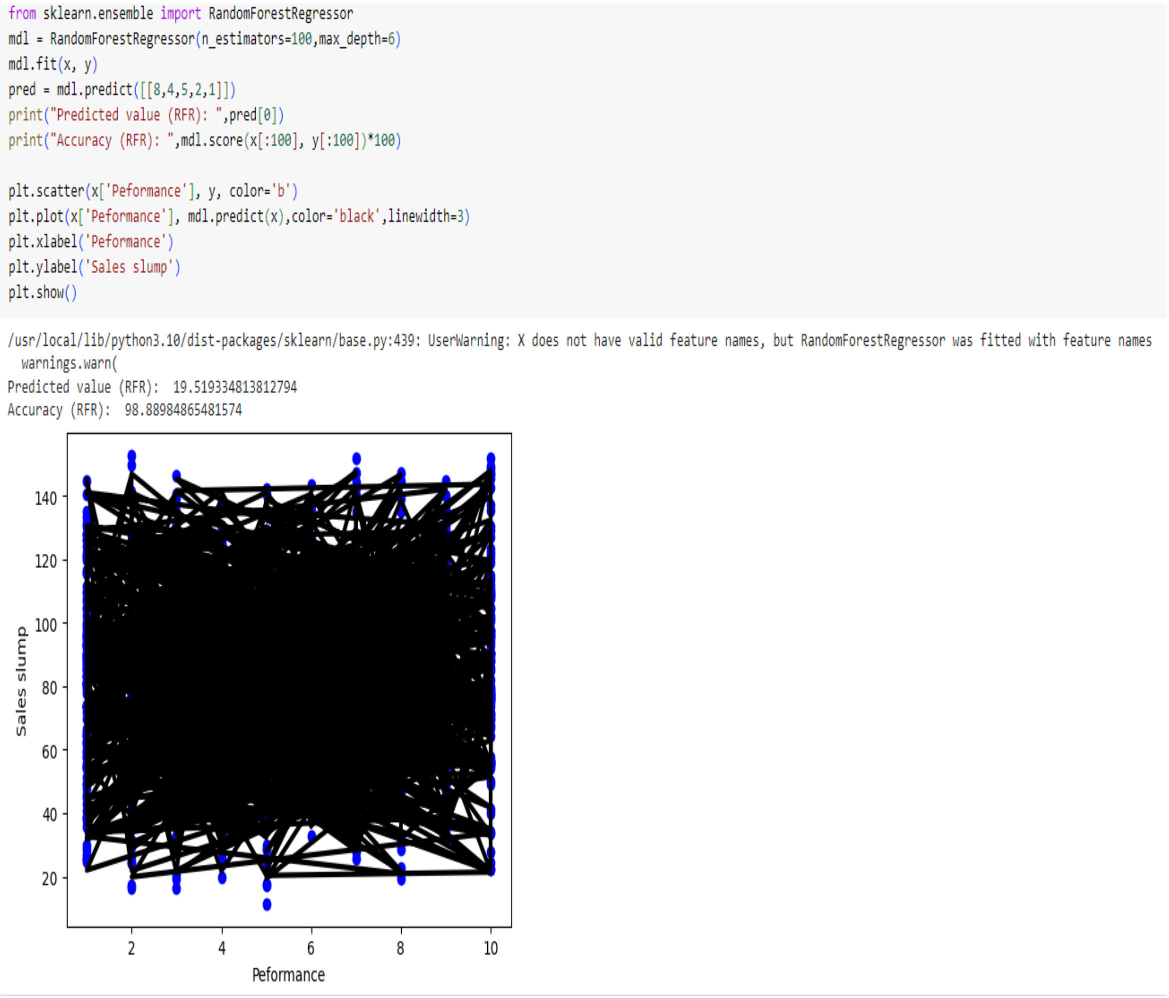
Output:



```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but SVR was fitted with feature names
  warnings.warn(
Predicted value (SVR):  34.74861349716815
Accuracy (SVR): 98.09874888288668
```

Random Forests:-

Random forests are a powerful and popular ensemble learning technique used for classification, regression, and anomaly detection. It is an extension of decision trees, where a large number of decision trees are trained on subsets of data. The final prediction is made by taking the average of all the individual tree predictions.



```
from sklearn.ensemble import RandomForestRegressor
mdl = RandomForestRegressor(n_estimators=100,max_depth=6)
mdl.fit(x, y)
pred = mdl.predict([[8,4,5,2,1]])
print("Predicted value (RFR): ",pred[0])
print("Accuracy (RFR): ",mdl.score(x[:100], y[:100])*100)

plt.scatter(x['Peformance'], y, color='b')
plt.plot(x['Peformance'], mdl.predict(x),color='black',linewidth=3)
plt.xlabel('Peformance')
plt.ylabel('Sales slump')
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
Predicted value (RFR):  19.519334813812794
Accuracy (RFR):  98.88984865481574
```

Evalution metrics

Evaluation metrics are used to evaluating machine learning models.We should know when to

use which metrics and it depends mainly on what kind of targets from sklearn import metrics.

Regression metrics:

Regression refers to predictive modeling problems that involve predicting a numeric value.It is

different from classification that involves predicting a class label. Unlike classification, you cannot

use classification accuracy to evaluate the predictions made by a regression model.

Mean squared error(MSE):

Mean Squared Error, or MSE for short, is a popular error metric for regression problems.It is

also an important loss function for algorithms fit or optimized using the least squares framing of a

regression problem. Here “least squares” refers to minimizing the mean squared error between

predictions and expected values.

MSE = 1 / N * sum for i to N (y_i – yhat_i)^2

Root Mean Squared Error:

The Root Mean Squared Error, or RMSE, is an extension of the mean squared error.Importantly,

the square root of the error is calculated, which means that the units of the RMSE are the same as

the original units of the target value that is being predicted.

Mean Absolute Error:

Mean Absolute Error, or MAE, is a popular metric because, like RMSE, the units of the error

score match the units of the target value that is being predicted.Unlike the RMSE, the changes in

MAE are linear and therefore intuitive.

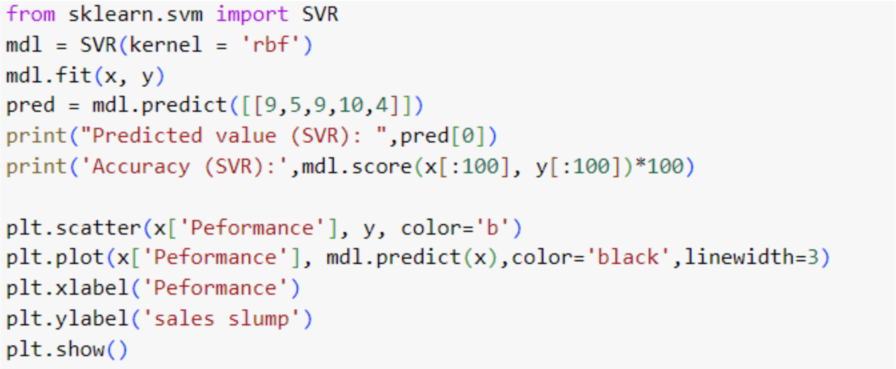
Support vector machines (SVM) :-

Support vector machines (SVM) are a supervised learning algorithm used for

classification and regression analysis. SVM tries to find the best hyperplane that separates the

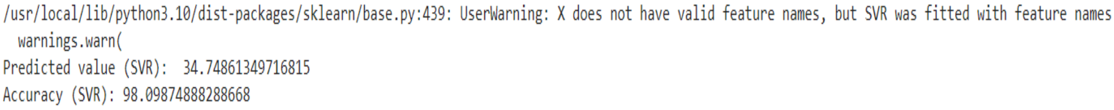
data points into different classes, with maximum margin. SVM can also handle non-linearly

separable data by transforming the data into a higher-dimensional space.

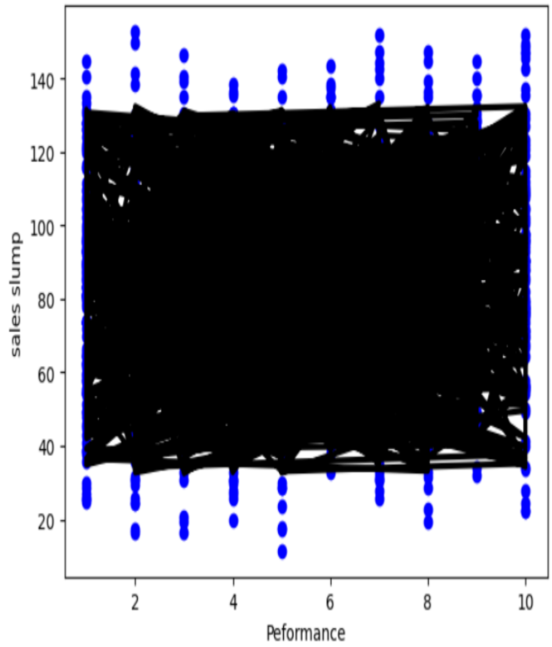


```
from sklearn.svm import SVR
mdl = SVR(kernel = 'rbf')
mdl.fit(x, y)
pred = mdl.predict([[9,5,9,10,4]])
print("Predicted value (SVR): ",pred[0])
print('Accuracy (SVR):',mdl.score(x[:100], y[:100])*100)

plt.scatter(x['Peformance'], y, color='b')
plt.plot(x['Peformance'], mdl.predict(x),color='black',linewidth=3)
plt.xlabel('Peformance')
plt.ylabel('sales slump')
plt.show()
```

Output:



```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but SVR was fitted with feature names
  warnings.warn(
Predicted value (SVR):  34.74861349716815
Accuracy (SVR): 98.09874888288668
```

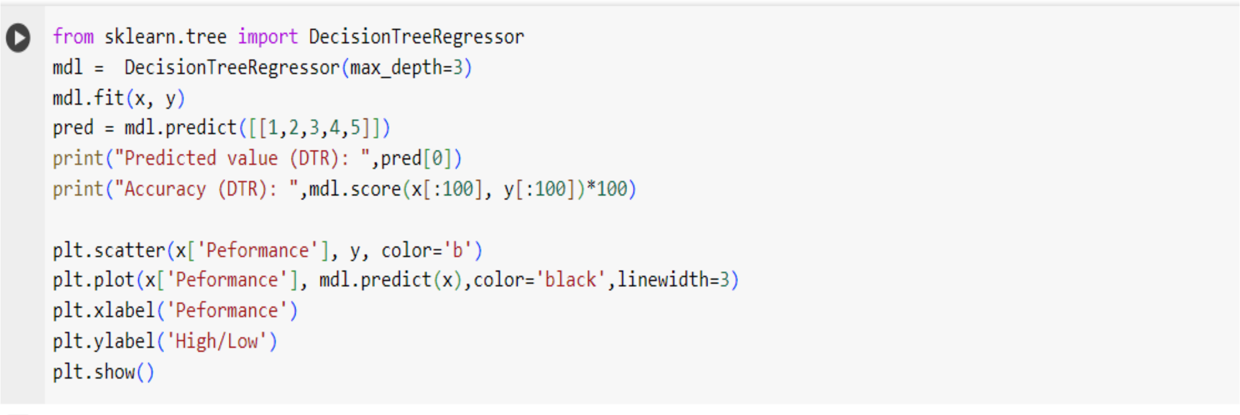


Decision Trees:-

A decision tree is a simple and intuitive model that is used for both

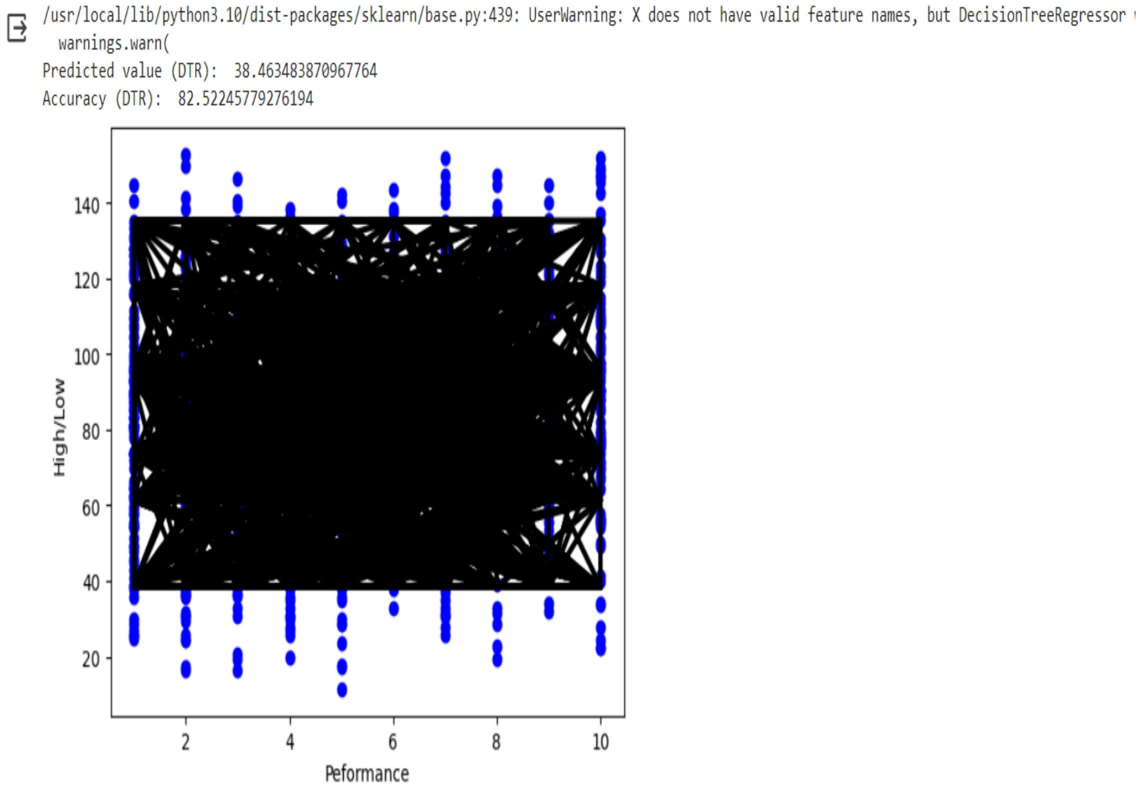
regression and classification problems. It is a tree-like structure where each node represents a feature or attribute, and each branch represents a decision rule.

Decision trees are particularly useful in situations where the data has multiple variables and is non-linear



```
from sklearn.tree import DecisionTreeRegressor
mdl =  DecisionTreeRegressor(max_depth=3)
mdl.fit(x, y)
pred = mdl.predict([[1,2,3,4,5]])
print("Predicted value (DTR): ",pred[0])
print("Accuracy (DTR): ",mdl.score(x[:100], y[:100])*100)

plt.scatter(x['Peformance'], y, color='b')
plt.plot(x['Peformance'], mdl.predict(x),color='black',linewidth=3)
plt.xlabel('Peformance')
plt.ylabel('High/Low')
plt.show()
```

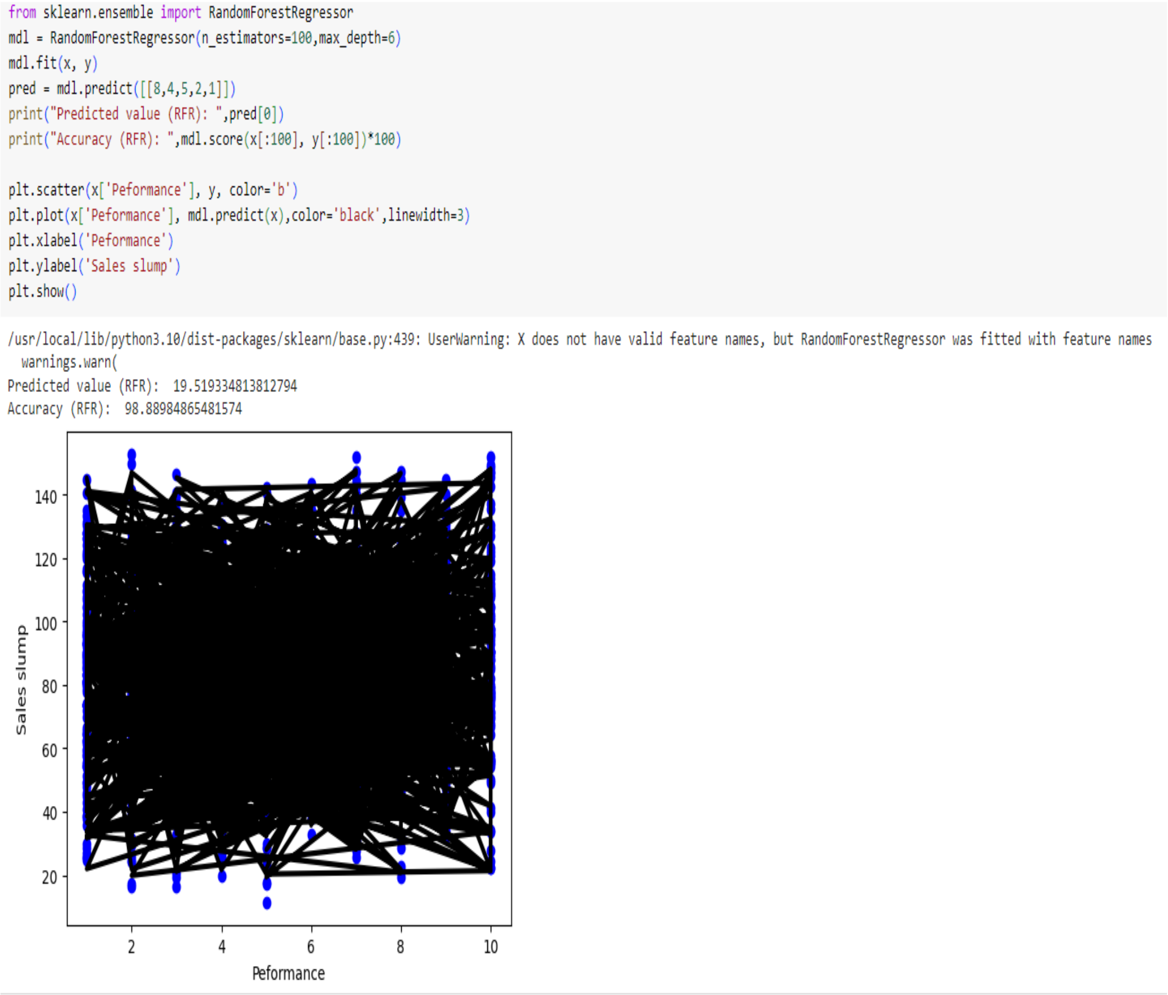
Output:



```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but DecisionTreeRegressor
  warnings.warn(
Predicted value (DTR):  38.463483870967764
Accuracy (DTR):  82.52245779276194
```

Random Forests:-

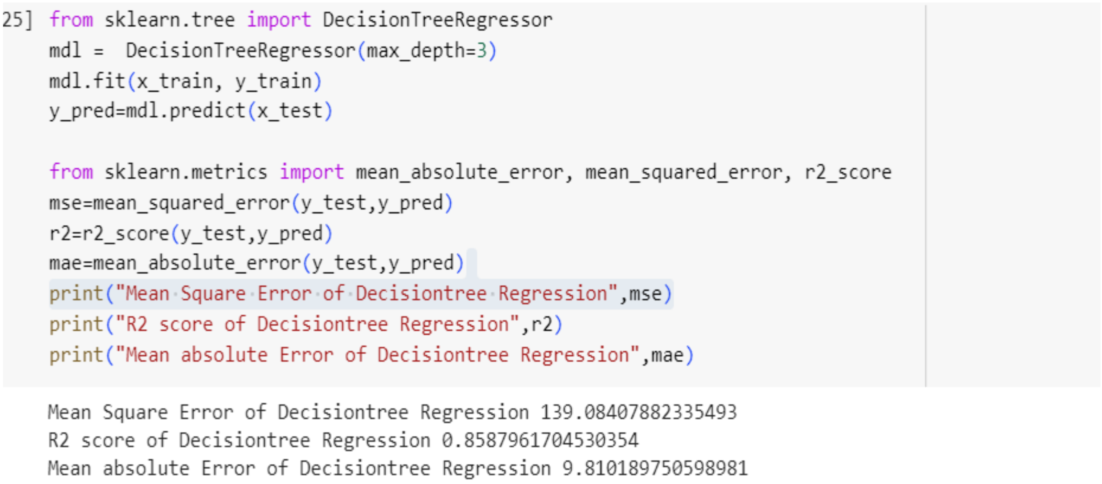
Random forests are a powerful and popular ensemble learning technique used for classification, regression, and anomaly detection. It is an extension of decision trees, where a large number of decision trees are trained on subsets of data. The final prediction is made by taking the average of all the individual tree predictions.



```
from sklearn.ensemble import RandomForestRegressor
mdl = RandomForestRegressor(n_estimators=100,max_depth=6)
mdl.fit(x, y)
pred = mdl.predict([[8,4,5,2,1]])
print("Predicted value (RFR): ",pred[0])
print("Accuracy (RFR): ",mdl.score(x[:100], y[:100])*100)

plt.scatter(x['Peformance'], y, color='b')
plt.plot(x['Peformance'], mdl.predict(x),color='black',linewidth=3)
plt.xlabel('Peformance')
plt.ylabel('Sales slump')
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
Predicted value (RFR):  19.519334813812794
Accuracy (RFR):  98.88984865481574
```

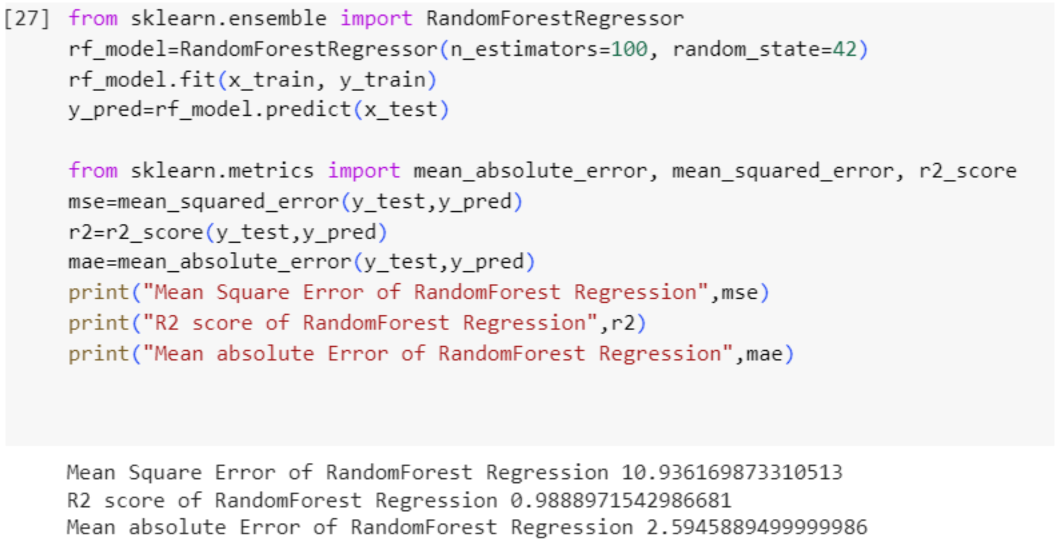
Decision tree error rating and accuracy.



```
25] from sklearn.tree import DecisionTreeRegressor
    mdl =  DecisionTreeRegressor(max_depth=3)
    mdl.fit(x_train, y_train)
    y_pred=mdl.predict(x_test)

    from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
    mse=mean_squared_error(y_test,y_pred)
    r2=r2_score(y_test,y_pred)
    mae=mean_absolute_error(y_test,y_pred)
    print("Mean Square Error of Decisiontree Regression",mse)
    print("R2 score of Decisiontree Regression",r2)
    print("Mean absolute Error of Decisiontree Regression",mae)
```

```
    Mean Square Error of Decisiontree Regression 139.08407882335493
    R2 score of Decisiontree Regression 0.8587961704530354
    Mean absolute Error of Decisiontree Regression 9.810189750598981
```

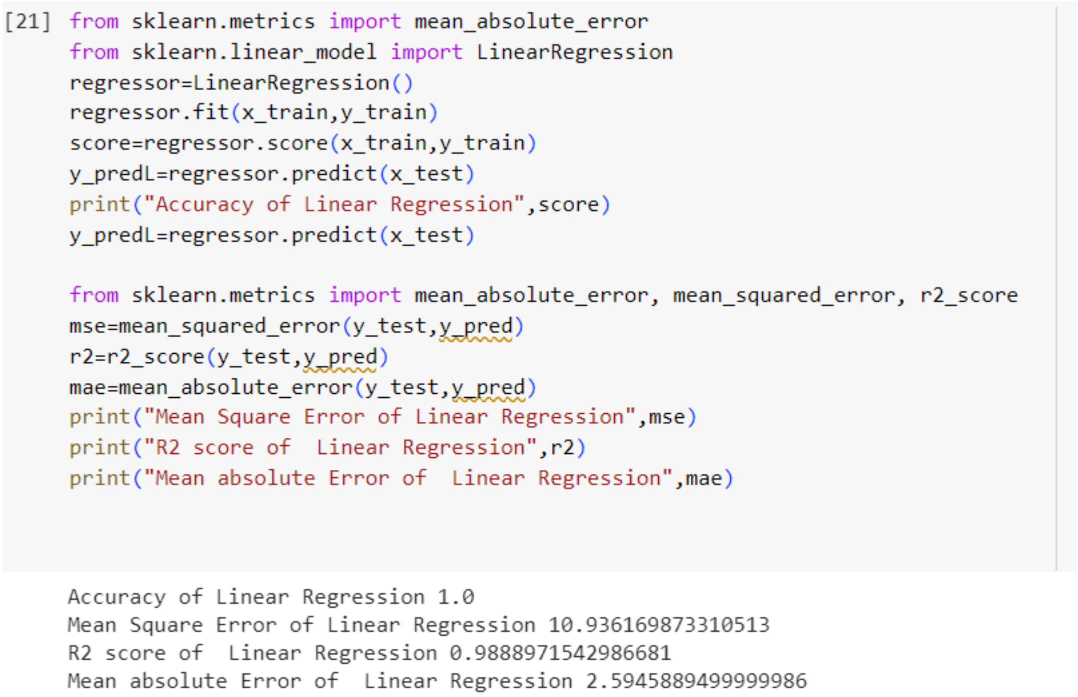
Random forest regression error rating and accuracy



```
[27] from sklearn.ensemble import RandomForestRegressor
     rf_model=RandomForestRegressor(n_estimators=100, random_state=42)
     rf_model.fit(x_train, y_train)
     y_pred=rf_model.predict(x_test)

     from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
     mse=mean_squared_error(y_test,y_pred)
     r2=r2_score(y_test,y_pred)
     mae=mean_absolute_error(y_test,y_pred)
     print("Mean Square Error of RandomForest Regression",mse)
     print("R2 score of RandomForest Regression",r2)
     print("Mean absolute Error of RandomForest Regression",mae)
```

```
Mean Square Error of RandomForest Regression 10.936169873310513
R2 score of RandomForest Regression 0.9888971542986681
Mean absolute Error of RandomForest Regression 2.5945889499999986
```

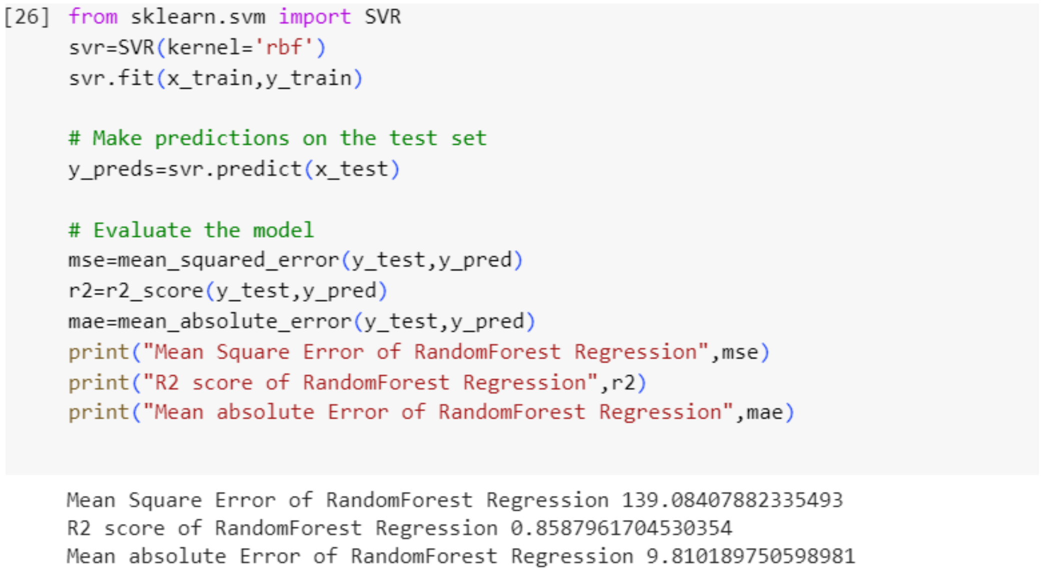
Linear regression error rating and accuracy



```
[21] from sklearn.metrics import mean_absolute_error
     from sklearn.linear_model import LinearRegression
     regressor=LinearRegression()
     regressor.fit(x_train,y_train)
     score=regressor.score(x_train,y_train)
     y_predL=regressor.predict(x_test)
     print("Accuracy of Linear Regression",score)
     y_predL=regressor.predict(x_test)

     from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
     mse=mean_squared_error(y_test,y_pred)
     r2=r2_score(y_test,y_pred)
     mae=mean_absolute_error(y_test,y_pred)
     print("Mean Square Error of Linear Regression",mse)
     print("R2 score of  Linear Regression",r2)
     print("Mean absolute Error of  Linear Regression",mae)
```

```
Accuracy of Linear Regression 1.0
Mean Square Error of Linear Regression 10.936169873310513
R2 score of  Linear Regression 0.9888971542986681
Mean absolute Error of  Linear Regression 2.5945889499999986
```

SVR error rating and accuracy



```
[26] from sklearn.svm import SVR
     svr=SVR(kernel='rbf')
     svr.fit(x_train,y_train)

     # Make predictions on the test set
     y_preds=svr.predict(x_test)

     # Evaluate the model
     mse=mean_squared_error(y_test,y_pred)
     r2=r2_score(y_test,y_pred)
     mae=mean_absolute_error(y_test,y_pred)
     print("Mean Square Error of RandomForest Regression",mse)
     print("R2 score of RandomForest Regression",r2)
     print("Mean absolute Error of RandomForest Regression",mae)
```

```
Mean Square Error of RandomForest Regression 139.08407882335493
R2 score of RandomForest Regression 0.8587961704530354
Mean absolute Error of RandomForest Regression 9.810189750598981
```

Conclusion:

Finally our model is decision tree by comparing with other four regressions which has low

error rating. So, our model is Decision tree.