EXPERIMENT 2.1

NAME- SECTION/GROUP –

UID- SEMESTER – 1ST

SUBJECT –DISRUPTIVE TECHNOLOGIES SUBJECT CODE –

BRANCH – CSE

Aim :

Develop a prediction model based on linear regression.

Tool Used:

Google Collab

Basic Concept/ Command Description:

Pycaret is a bundle of many Machine Learning algorithms. Only three lines of code is required to compare 20 ML models.

Pycaret is available for:

Classification Regression Clustering

Code and Observations, Simulation Screenshots and Discussions:

```
[4] !pip install pycaret &> /dev/null
    print("Pycaret installed successfully!")

    Pycaret installed successfully!
```

Install Pycaret
#Get the version

```
from pycaret.utils import version
version()
```

```
'2.3.5'
```

#Loading dataset from Pycaret

```
[6]  from pycaret.datasets import get_data
```

#Get the list of datasets available in pycaret

```
dataSets = get_data('index')
```

	Dataset	Data Types	Default Task	Target Variable 1	Target Variable 2	# Instances	# Attributes	Missing Values
0	anomaly	Multivariate	Anomaly Detection	None	None	1000	10	N
1	france	Multivariate	Association Rule Mining	InvoiceNo	Description	8557	8	N
2	germany	Multivariate	Association Rule Mining	InvoiceNo	Description	9495	8	N
3	bank	Multivariate	Classification (Binary)	deposit	None	45211	17	N
4	blood	Multivariate	Classification (Binary)	Class	None	748	5	N
5	cancer	Multivariate	Classification (Binary)	Class	None	683	10	N
6	credit	Multivariate	Classification (Binary)	default	None	24000	24	N
7	diabetes	Multivariate	Classification (Binary)	Class variable	None	768	9	N
8	electrical_grid	Multivariate	Classification (Binary)	stabf	None	10000	14	N
9	employee	Multivariate	Classification (Binary)	left	None	14999	10	N
10	heart	Multivariate	Classification (Binary)	DEATH	None	200	16	N
11	heart_disease	Multivariate	Classification (Binary)	Disease	None	270	14	N
12	hepatitis	Multivariate	Classification (Binary)	Class	None	154	32	Y

#Get boston dataset

```
bostonDataSet=get_data("boston")
```

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2

#Get the shape of Boston Dataset

```
[7] bostonDataSet.shape
```

```
(506, 14)
```

#CREATE ENVIRONMENT SETUP

```
[11] from pycaret.regression import * #create environment setup
```

#INITIALIZATION OF SETUP

```
exp_reg = setup(bostonDataSet , target='medv')
```

	Description	Value
0	session_id	6616
1	Target	medv
2	Original Data	(506, 14)
3	Missing Values	False
4	Numeric Features	11
5	Categorical Features	2
6	Ordinal Features	False
7	High Cardinality Features	False
8	High Cardinality Method	None
9	Transformed Train Set	(354, 21)
10	Transformed Test Set	(152, 21)
11	Shuffle Train-Test	True
12	Stratify Train-Test	False
13	Fold Generator	KFold
14	Fold Number	10

#Compare Model

```
[14] cm=compare_models()
```

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
et	Extra Trees Regressor	2.1229	10.7628	3.1342	0.8610	0.1355	0.1053	0.449
gbr	Gradient Boosting Regressor	2.1692	11.6619	3.2899	0.8407	0.1448	0.1099	0.104
lightgbm	Light Gradient Boosting Machine	2.3620	12.3738	3.4325	0.8285	0.1454	0.1145	0.078
rf	Random Forest Regressor	2.3592	14.5993	3.6644	0.8087	0.1561	0.1181	0.538
ada	AdaBoost Regressor	2.7228	16.0411	3.8793	0.7899	0.1715	0.1409	0.105
lr	Linear Regression	3.2717	21.0186	4.5431	0.7098	0.2174	0.1588	0.318
ridge	Ridge Regression	3.2493	21.0439	4.5467	0.7096	0.2207	0.1578	0.015
br	Bayesian Ridge	3.2226	21.2267	4.5664	0.7069	0.2222	0.1565	0.018
en	Elastic Net	3.5074	25.7807	5.0053	0.6555	0.2247	0.1642	0.017
lasso	Lasso Regression	3.5446	26.2378	5.0442	0.6514	0.2275	0.1664	0.017
huber	Huber Regressor	3.7149	29.2137	5.2580	0.6251	0.2775	0.1785	0.051
omp	Orthogonal Matching Pursuit	3.8469	27.9578	5.2256	0.6240	0.3077	0.1991	0.015
dt	Decision Tree Regressor	3.2754	33.3368	5.5594	0.5658	0.2263	0.1722	0.021
knn	K Neighbors Regressor	4.5338	43.6502	6.4799	0.4166	0.2464	0.2053	0.066
lar	Least Angle Regression	4.1666	41.6927	5.6998	0.3233	0.2892	0.1981	0.018

#Build a single model Random Forest

```
[16] rfmodel = create_model('rf')
```

	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	2.5150	10.3462	3.2165	0.8679	0.1507	0.1209
1	2.7944	15.7432	3.9678	0.8017	0.2074	0.1588
2	2.0890	6.6850	2.5855	0.7323	0.1347	0.1045
3	2.7392	25.8642	5.0857	0.6580	0.1814	0.1422
4	2.3411	18.0320	4.2464	0.7405	0.1630	0.1226
5	1.4134	4.1827	2.0452	0.9482	0.0890	0.0654
6	2.6111	19.8139	4.4513	0.8344	0.1889	0.1340
7	2.8459	15.0071	3.8739	0.8247	0.1742	0.1388
8	1.5753	4.2832	2.0696	0.9192	0.1103	0.0889
9	2.6676	26.0354	5.1025	0.7598	0.1609	0.1046
Mean	2.3592	14.5993	3.6644	0.8087	0.1561	0.1181
SD	0.4833	7.7057	1.0822	0.0847	0.0344	0.0264

#Save Model

```
[17] sm=save_model(rfmodel,'rmodelfile')
```

```
Transformation Pipeline and Model Successfully Saved
```

#Load Model

```
[18] rfmodel=load_model('rmodelfile')
```

```
Transformation Pipeline and Model Successfully Loaded
```

#New Predictions

```
[19] newdataset=get_data("boston").iloc[:5]
```

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2

```
[20] newpredictions=predict_model(rfmodel,data=newdataset)
     newpredictions
```

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv	Label
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0	25.719000
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6	22.389000
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7	35.005001
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4	34.935001
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2	35.801001

#Plotting the comparison of actual and predicted model

```
[ ] import matplotlib.pyplot as plt
```

```
predicted=newpredictions.iloc[:,-1]
actual=newpredictions.iloc[:,-2]
plt.scatter(actual,predicted)
plt.xlabel('predicted')
plt.ylabel('actual')
plt.title('actual Vs predicted')
plt.savefig("result.jpg", dpi=300)
plt.show()
```

MODEL