University Institute of Engineering

Department of Computer Science & Engineering

Experiment : 3

Student Name: Animesh Kumar **UID:22BCS13257**

Branch: Computer Science & Engineering **Section/Group:22CSE-212/C**

Semester: 1st **Date of Performance:09/11/2022**

Subject Name: Disruptive Technology-1

Subject Code: 22ECH-102

1. **Aim of the practical:** Explore,visualize,transform and summarize input datasets for building Classificatio/regression/prediction models.
2. **Tool Used:** Google Colaboratory
3. **Basic Concept/ Command Description:**

- It 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

4. **Code:**

(a) Install Pycaret

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

```
Pycaret installed sucessfully!!
```

(b) Get the version of the pycaret

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

```
'2.3.10'
```

1. Regression: Basics

1.1 Loading Dataset - Loading dataset from pycaret

```
[ ] from pycaret.datasets import get_data

    # No output
```

1.2 Get the list of datasets available in pycaret (55)

```
[ ] # Internet connection is required
    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

1.3 Get boston dataset

```
bostonDataSet = get_data("boston")    # SN is 46
# This is regression dataset. The values in medv are continuous values
```

	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

Read data from file

```
[ ] # import pandas as pd
    # bostonDataSet = pd.read_csv("myFile.csv")
```

1.4 Parameter setting for all regression models

- Train/Test division
- Sampling
- Normalization
- Transformation
- PCA (Dimention Reduction)
- Handaling of Outliers
- Feature Selection

```
[ ] from pycaret.regression import *
    s = setup(data =  bostonDataSet, target='medv', silent=True)
```

	Description	Value
0	session_id	2660

1.4 Parameter setting for all regression models

- Train/Test division
- Sampling
- Normalization
- Transformation
- PCA (Dimention Reduction)
- Handaling of Outliers
- Feature Selection

```
[ ] from pycaret.regression import *
    s = setup(data =  bostonDataSet, target='me
```

	Description
0	session_id
1	Target
2	Original Data
3	Missing Values
4	Numeric Features
5	Categorical Features
6	Ordinal Features
7	High Cardinality Features
8	High Cardinality Method
9	Transformed Train Set
10	Transformed Test Set
11	Shuffle Train-Test
12	Stratify Train-Test
13	Fold Generator
14	Fold Number

1.5 Run and compare the Model Performance

```
cm = compare_models()
# Explore more parameters
```

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
gbr	Gradient Boosting Regressor	2.2709	11.6556	3.3222	0.8598	0.1436	0.1115	0.207
et	Extra Trees Regressor	2.2554	12.4863	3.4064	0.8503	0.1458	0.1104	0.445
lightgbm	Light Gradient Boosting Machine	2.3430	13.4238	3.5675	0.8377	0.1519	0.1144	0.100
rf	Random Forest Regressor	2.3600	15.0882	3.7744	0.8220	0.1550	0.1145	0.515
ada	AdaBoost Regressor	2.7781	16.9579	4.0097	0.7963	0.1797	0.1444	0.163
lr	Linear Regression	3.4844	25.0307	4.8881	0.7050	0.2342	0.1719	0.296
ridge	Ridge Regression	3.4615	24.9773	4.8760	0.7048	0.2437	0.1718	0.011
br	Bayesian Ridge	3.4810	25.2439	4.9117	0.7013	0.2396	0.1730	0.012
dt	Decision Tree Regressor	3.2041	25.3979	4.8884	0.7012	0.2047	0.1524	0.017
huber	Huber Regressor	3.6121	29.0657	5.2986	0.6605	0.2636	0.1796	0.038
lasso	Lasso Regression	3.8375	29.5854	5.3890	0.6543	0.2554	0.1808	0.014
en	Elastic Net	3.8999	30.1773	5.4505	0.6466	0.2553	0.1826	0.013
lar	Least Angle Regression	3.8958	31.6155	5.4408	0.6283	0.2610	0.1911	0.016
omp	Orthogonal Matching Pursuit	3.9588	36.9392	5.9396	0.5627	0.3255	0.2085	0.012
knn	K Neighbors Regressor	4.5305	44.9704	6.6194	0.4937	0.2506	0.2069	0.060
par	Passive Aggressive Regressor	6.7563	83.2167	9.0684	0.0380	0.4436	0.3468	0.012
llar	Lasso Least Angle Regression	6.8483	88.6514	9.3716	-0.0132	0.3921	0.3630	0.012
dummy	Dummy Regressor	6.8483	88.6514	9.3716	-0.0132	0.3921	0.3630	0.009

2. Regression: Advance - 1

2.1 Model Performance using data "Normalization"

```
s = setup(data = bostonDataSet, target = 'medv', normalize = True, normalize_method = 'zscore', silent=True)
cm = compare_models()

#normalize_method = {zscore, minmax, maxabs, robust}
```

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
et	Extra Trees Regressor	2.1255	10.0518	3.0285	0.8804	0.1366	0.1059	0.438
gbr	Gradient Boosting Regressor	2.3016	11.9015	3.2632	0.8570	0.1495	0.1162	0.098
rf	Random Forest Regressor	2.3353	12.8774	3.3679	0.8465	0.1462	0.1145	0.508
lightgbm	Light Gradient Boosting Machine	2.4766	13.2894	3.4968	0.8400	0.1555	0.1240	0.037
ada	AdaBoost Regressor	2.9089	16.5975	3.9249	0.8032	0.1824	0.1528	0.086
knn	K Neighbors Regressor	2.7686	18.7025	4.0769	0.7815	0.1701	0.1315	0.060
dt	Decision Tree Regressor	3.3548	23.9393	4.7495	0.7134	0.1980	0.1581	0.016
br	Bayesian Ridge	3.3849	24.1437	4.8156	0.7050	0.2537	0.1694	0.012
ridge	Ridge Regression	3.4327	24.1599	4.8249	0.7039	0.2499	0.1720	0.012
lr	Linear Regression	3.4545	24.2222	4.8338	0.7029	0.2489	0.1732	0.012
huber	Huber Regressor	3.2291	25.7840	4.9211	0.6896	0.2712	0.1575	0.025
lar	Least Angle Regression	3.9771	30.3047	5.4146	0.6344	0.2775	0.1969	0.017
lasso	Lasso Regression	3.8038	30.4733	5.4145	0.6338	0.2747	0.1942	0.013
en	Elastic Net	3.8715	31.7548	5.5356	0.6197	0.2421	0.1929	0.012
omp	Orthogonal Matching Pursuit	4.4165	41.1317	6.2387	0.5083	0.3080	0.2192	0.012
par	Passive Aggressive Regressor	4.7499	43.9339	6.3795	0.4757	0.3755	0.2557	0.013
llar	Lasso Least Angle Regression	6.6376	84.0019	9.1265	-0.0180	0.3882	0.3613	0.013
dummy	Dummy Regressor	6.6376	84.0019	9.1265	-0.0180	0.3882	0.3613	0.010

2.2 Model Performance using "Feature Selection"

```
s = setup(data = bostonDataSet, target = 'medv', feature_selection = True, feature_selection_threshold = 0.9, silent=True)
cm = compare_models()
```

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
et	Extra Trees Regressor	2.2333	11.5437	3.2768	0.8404	0.1501	0.1150	0.444
gbr	Gradient Boosting Regressor	2.2487	11.6528	3.2693	0.8366	0.1531	0.1170	0.097
rf	Random Forest Regressor	2.3575	13.5977	3.5407	0.8144	0.1580	0.1202	0.510
lightgbm	Light Gradient Boosting Machine	2.4635	13.6660	3.5955	0.8121	0.1621	0.1258	0.039
ada	AdaBoost Regressor	2.7269	16.6389	3.9934	0.7680	0.1804	0.1436	0.093
dt	Decision Tree Regressor	3.0496	21.5964	4.5034	0.7143	0.2070	0.1587	0.017
lr	Linear Regression	3.4893	25.6676	4.9208	0.6717	0.2527	0.1763	0.013
ridge	Ridge Regression	3.4693	25.8383	4.9294	0.6715	0.2636	0.1767	0.014
br	Bayesian Ridge	3.5185	26.6405	5.0119	0.6618	0.2703	0.1793	0.014
en	Elastic Net	3.8338	31.0114	5.4287	0.6055	0.2758	0.1839	0.014
lasso	Lasso Regression	3.8607	31.5612	5.4718	0.5989	0.2809	0.1839	0.014
huber	Huber Regressor	3.7851	32.5452	5.5259	0.5783	0.2772	0.1885	0.037
omp	Orthogonal Matching Pursuit	4.1148	34.2298	5.7521	0.5504	0.3106	0.2048	0.013
knn	K Neighbors Regressor	4.6954	43.2935	6.4667	0.4146	0.2562	0.2230	0.061
llar	Lasso Least Angle Regression	6.4531	81.7067	8.9189	-0.0587	0.3812	0.3544	0.013
par	Passive Aggressive Regressor	7.9590	108.8463	10.2679	-0.5595	0.4619	0.4403	0.017
lar	Least Angle Regression	40.8642	23594.6299	53.4746	-209.9022	0.5292	2.6892	0.017

2.3 Model Performance using "Outlier Removal"

```
s = setup(data = bostonDataSet, target = 'medv', remove_outliers = True, outliers_threshold = 0.05, silent=True)
cm = compare_models()
```

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
et	Extra Trees Regressor	1.9469	7.9527	2.7527	0.8827	0.1265	0.0990	0.451
gbr	Gradient Boosting Regressor	2.0667	9.4401	2.8955	0.8661	0.1373	0.1068	0.091
rf	Random Forest Regressor	2.1141	9.9440	2.9692	0.8575	0.1396	0.1104	0.509
lightgbm	Light Gradient Boosting Machine	2.1797	10.6409	3.0521	0.8502	0.1348	0.1090	0.039
ada	AdaBoost Regressor	2.5725	13.3359	3.5250	0.8064	0.1628	0.1339	0.092
dt	Decision Tree Regressor	2.8802	17.5934	3.9450	0.7490	0.1889	0.1490	0.016
ridge	Ridge Regression	2.9756	17.5036	4.0487	0.7473	0.2205	0.1533	0.013
lr	Linear Regression	2.9833	17.5430	4.0532	0.7469	0.2225	0.1538	0.013
br	Bayesian Ridge	2.9701	17.6438	4.0684	0.7445	0.2147	0.1524	0.014
lar	Least Angle Regression	3.2810	21.4263	4.4838	0.6764	0.2442	0.1690	0.017
huber	Huber Regressor	3.3111	22.4529	4.6078	0.6679	0.2342	0.1714	0.037
en	Elastic Net	3.4360	23.2754	4.7230	0.6602	0.2073	0.1648	0.015
lasso	Lasso Regression	3.4478	23.5117	4.7435	0.6586	0.2077	0.1651	0.013
omp	Orthogonal Matching Pursuit	3.5700	24.1500	4.8044	0.6489	0.2734	0.1906	0.014
knn	K Neighbors Regressor	4.2713	38.7975	6.1020	0.4348	0.2409	0.2012	0.063
llar	Lasso Least Angle Regression	6.0449	70.8467	8.3099	-0.0305	0.3665	0.3354	0.013
par	Passive Aggressive Regressor	8.1154	101.2752	9.8986	-0.7237	0.4800	0.4512	0.016

2.4 Model Performance using "Transformation"

```
s = setup(data = bostonDataSet, target = 'medv', transformation = True, transformation_method = 'yeo-johnson', silent=True)
cm = compare_models()
```

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
gbr	Gradient Boosting Regressor	2.1953	8.3473	2.8497	0.8819	0.1444	0.1161	0.092
rf	Random Forest Regressor	2.2168	9.1829	2.9570	0.8795	0.1439	0.1163	0.501
et	Extra Trees Regressor	2.1738	9.8099	2.9943	0.8766	0.1379	0.1100	0.450
ada	AdaBoost Regressor	2.5934	11.0381	3.2612	0.8468	0.1688	0.1419	0.094
lightgbm	Light Gradient Boosting Machine	2.4271	12.1036	3.3776	0.8467	0.1595	0.1257	0.040
knn	K Neighbors Regressor	2.8261	17.2318	4.0407	0.7684	0.1809	0.1419	0.062
dt	Decision Tree Regressor	2.8962	16.6474	3.9897	0.7658	0.1931	0.1523	0.015
br	Bayesian Ridge	3.3157	20.4448	4.4028	0.7488	0.2177	0.1746	0.014
ridge	Ridge Regression	3.3548	20.5828	4.4250	0.7463	0.2192	0.1765	0.014
lar	Least Angle Regression	3.3750	20.6475	4.4360	0.7450	0.2200	0.1774	0.017
lr	Linear Regression	3.3746	20.6844	4.4387	0.7447	0.2200	0.1774	0.014
huber	Huber Regressor	3.2126	22.7493	4.5813	0.7166	0.2289	0.1662	0.025
lasso	Lasso Regression	3.6192	26.8434	4.9877	0.6847	0.2254	0.1824	0.016
en	Elastic Net	3.7126	30.1710	5.2528	0.6543	0.2326	0.1939	0.014
omp	Orthogonal Matching Pursuit	4.0011	28.6006	5.2356	0.6380	0.2676	0.2016	0.013
par	Passive Aggressive Regressor	4.2995	35.5631	5.8078	0.5611	0.2982	0.2297	0.019
llar	Lasso Least Angle Regression	6.7468	86.4376	9.1104	-0.0422	0.3946	0.3740	0.013

2.5 Model Performance using "PCA"

```
s = setup(data = bostonDataSet, target = 'medv', pca = True, pca_method = 'linear', silent=True)
cm = compare_models()
```

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
rf	Random Forest Regressor	4.7579	50.1766	6.9202	0.3842	0.2812	0.2364	0.434
lightgbm	Light Gradient Boosting Machine	4.8950	50.6940	6.9594	0.3747	0.2826	0.2395	0.030
gbr	Gradient Boosting Regressor	4.7525	52.1176	6.9834	0.3685	0.2827	0.2365	0.058
knn	K Neighbors Regressor	4.8988	50.4595	7.0002	0.3616	0.2829	0.2418	0.061
et	Extra Trees Regressor	4.8876	54.2282	7.1564	0.3354	0.2892	0.2445	0.409
lr	Linear Regression	5.3394	58.2617	7.4706	0.2877	0.2995	0.2579	0.014
ridge	Ridge Regression	5.3394	58.2617	7.4706	0.2877	0.2995	0.2579	0.013
lar	Least Angle Regression	5.3394	58.2617	7.4706	0.2877	0.2995	0.2579	0.014
lasso	Lasso Regression	5.3410	58.2647	7.4711	0.2876	0.2996	0.2580	0.015
en	Elastic Net	5.3403	58.2628	7.4709	0.2876	0.2995	0.2580	0.014
br	Bayesian Ridge	5.3625	58.3582	7.4803	0.2858	0.3003	0.2596	0.013
huber	Huber Regressor	4.9728	60.5764	7.5819	0.2721	0.2950	0.2249	0.023
omp	Orthogonal Matching Pursuit	5.8238	63.8401	7.8589	0.2081	0.3176	0.2838	0.013
ada	AdaBoost Regressor	5.9631	64.3540	7.8959	0.1861	0.3242	0.3067	0.029
dt	Decision Tree Regressor	5.8178	83.6042	8.8239	-0.0306	0.3503	0.2875	0.014
llar	Lasso Least Angle Regression	6.5957	83.0052	9.0027	-0.0382	0.3860	0.3576	0.015
par	Passive Aggressive Regressor	16.2110	589.6839	23.0010	-7.6021	0.7030	0.8759	0.016

2.6 Model Performance using "Outlier Removal" + "Normalization"

```
s = setup(data = bostonDataSet, target = 'medv', remove_outliers = True, outliers_threshold = 0.05, normalize = True, normalize_method = 'zscore', silent=True)
cm = compare_models()
```

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
et	Extra Trees Regressor	2.0944	9.5989	2.9828	0.8770	0.1411	0.1076	0.449
gbr	Gradient Boosting Regressor	2.2069	9.8574	3.0758	0.8706	0.1483	0.1155	0.091
rf	Random Forest Regressor	2.2991	11.6036	3.2677	0.8537	0.1533	0.1187	0.500
lightgbm	Light Gradient Boosting Machine	2.4453	12.5609	3.4327	0.8446	0.1595	0.1251	0.039
ada	AdaBoost Regressor	2.7620	14.8197	3.7711	0.8060	0.1834	0.1507	0.093
dt	Decision Tree Regressor	3.1009	19.4040	4.2898	0.7403	0.2175	0.1684	0.017
knn	K Neighbors Regressor	3.0696	23.4017	4.5924	0.7349	0.1794	0.1380	0.061
br	Bayesian Ridge	3.4865	24.7083	4.8472	0.6969	0.2462	0.1742	0.013
ridge	Ridge Regression	3.5338	24.7467	4.8591	0.6946	0.2504	0.1764	0.014
lr	Linear Regression	3.5521	24.8497	4.8733	0.6925	0.2530	0.1772	0.012
lar	Least Angle Regression	3.6679	25.3953	4.9147	0.6875	0.2553	0.1836	0.016
huber	Huber Regressor	3.2997	27.3000	5.0243	0.6602	0.2729	0.1616	0.027
lasso	Lasso Regression	3.8353	31.2229	5.4442	0.6330	0.2582	0.1946	0.016
en	Elastic Net	3.9348	33.6859	5.6159	0.6153	0.2447	0.1982	0.014
omp	Orthogonal Matching Pursuit	4.0298	32.5366	5.6006	0.5956	0.2891	0.2094	0.014
par	Passive Aggressive Regressor	4.9774	44.8913	6.5955	0.4387	0.4272	0.2604	0.014
llar	Lasso Least Angle Regression	6.6842	86.5454	9.1356	-0.0080	0.3944	0.3687	0.013

2.7 Model Performance using "Outlier Removal" + "Normalization" + "Transformation"

```
s = setup(data = bostonDataSet, target = 'medv', remove_outliers = True, outliers_threshold = 0.05, normalize = True, normalize_method = 'zscore', transformation = True, transformation_method = 'yeo-johnson', silent=True)
cm = compare_models()
```

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
et	Extra Trees Regressor	2.0459	9.2277	2.9296	0.8554	0.1313	0.1032	0.444
gbr	Gradient Boosting Regressor	2.1264	10.3774	3.1065	0.8288	0.1449	0.1107	0.090
rf	Random Forest Regressor	2.2446	11.2221	3.2544	0.8208	0.1440	0.1127	0.511
lightgbm	Light Gradient Boosting Machine	2.3255	12.0976	3.3920	0.8115	0.1427	0.1117	0.038
ada	AdaBoost Regressor	2.8007	15.4455	3.8236	0.7639	0.1768	0.1490	0.097
knn	K Neighbors Regressor	2.8570	21.1692	4.4358	0.7001	0.1781	0.1345	0.062
ridge	Ridge Regression	3.3023	21.1415	4.5220	0.6801	0.2091	0.1674	0.014
br	Bayesian Ridge	3.2681	21.2289	4.5233	0.6799	0.2084	0.1659	0.014
lr	Linear Regression	3.3208	21.2061	4.5312	0.6786	0.2101	0.1683	0.015
huber	Huber Regressor	3.0957	22.4442	4.5891	0.6708	0.2079	0.1519	0.025
lar	Least Angle Regression	3.4288	22.2781	4.6308	0.6679	0.2347	0.1753	0.017
dt	Decision Tree Regressor	2.9408	19.5589	4.1751	0.6455	0.1949	0.1511	0.019
lasso	Lasso Regression	3.4574	25.6062	4.9540	0.6282	0.2144	0.1743	0.015
en	Elastic Net	3.5328	27.7193	5.1256	0.6003	0.2237	0.1828	0.014
omp	Orthogonal Matching Pursuit	3.7297	27.4329	5.1626	0.5834	0.2236	0.1796	0.012
par	Passive Aggressive Regressor	4.1073	32.8640	5.6240	0.4674	0.2719	0.2093	0.015
llar	Lasso Least Angle Regression	5.9734	70.9945	8.2995	-0.0256	0.3698	0.3372	0.013

3. Regression: Advance - 2

3.1 Build a single model - "RandomForest"

```
from pycaret.datasets import get_data
from pycaret.regression import *

bostonDataSet = get_data("boston")    # SN is 46
s = setup(data =  bostonDataSet, target='medv', silent=True)

gbrModel = create_model('gbr')
# Explore more parameters
```

Fold	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	3.0147	27.9895	5.2905	0.6670	0.2050	0.1256
1	2.1475	9.4255	3.0701	0.8943	0.1223	0.0940
2	2.6840	15.9089	3.9886	0.8016	0.1844	0.1337
3	1.9261	7.1942	2.6822	0.9136	0.1190	0.1013
4	2.0934	7.3000	2.7019	0.9277	0.1372	0.1104
5	1.8604	5.9163	2.4323	0.9402	0.1022	0.0943
6	2.0981	6.5233	2.5541	0.9114	0.1355	0.1019
7	1.8001	4.6829	2.1640	0.9276	0.1170	0.0979
8	2.3522	11.0896	3.3301	0.8352	0.1932	0.1613
9	2.9115	24.4266	4.9423	0.7415	0.1989	0.1521
Mean	2.2888	12.0457	3.3156	0.8560	0.1515	0.1173
Std	0.4151	7.7365	1.0259	0.0880	0.0373	0.0234

3.3 Save the trained model

```
sm = save_model(gbrModel, 'gbrModelFile')
```

```
INFO:logs:save_model() successfully completed......................................
Transformation Pipeline and Model Successfully Saved
```

3.4 Load the model

```
[ ] gbrModel = load_model('gbrModelFile')
```

```
INFO:logs:Initializing load_model()
INFO:logs:load_model(model_name=gbrModelFile, platform=None, authentication=None, verbose=True)
Transformation Pipeline and Model Successfully Loaded
```

3.5 Make prediction on the new dataset

Get new dataset

```
# Select top 10 rows from boston dataset
newDataSet = get_data("boston").iloc[:10]
```

	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

Make prediction on new dataset

```
[ ] newPredictions = predict_model(gbrModel, data = newDataSet)
    newPredictions
```

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	Gradient Boosting Regressor	2.2571	12.5487	3.5424	0.6976	0.1568	0.0907

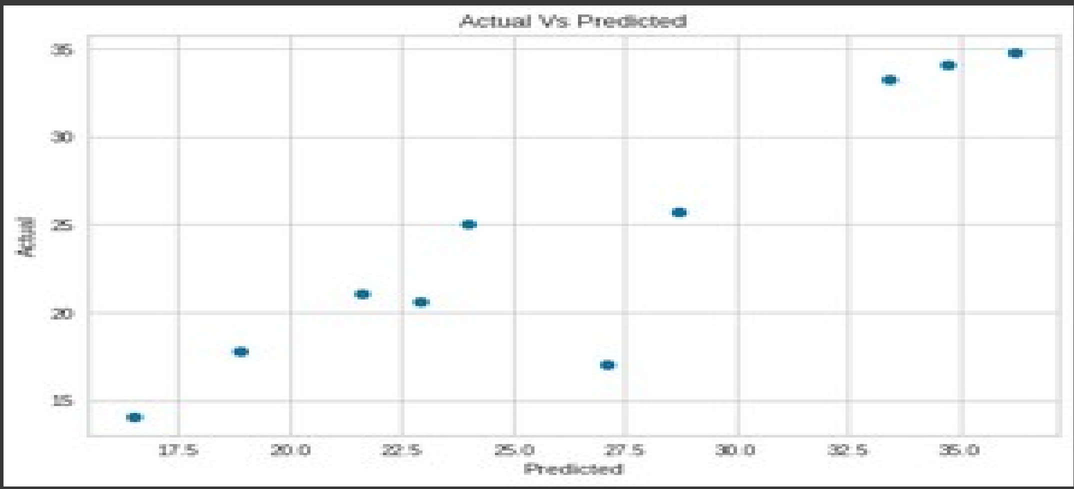
	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv	Label
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0	25.057511
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6	21.103533
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7	34.061129
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	33.233713
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	34.752324
5	0.02985	0.0	2.18	0	0.458	6.430	58.7	6.0622	3	222	18.7	394.12	5.21	28.7	25.752128
6	0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2	395.60	12.43	22.9	20.623648
7	0.14455	12.5	7.87	0	0.524	6.172	96.1	5.9505	5	311	15.2	396.90	19.15	27.1	17.070027
8	0.21124	12.5	7.87	0	0.524	5.631	100.0	6.0821	5	311	15.2	386.63	29.93	16.5	14.053086
9	0.17004	12.5	7.87	0	0.524	6.004	85.9	6.5921	5	311	15.2	386.71	17.10	18.9	17.836429

3.6 Scatter plot b/w actual and predicted

```
import matplotlib.pyplot as plt

predicted = newPredictions.iloc[:,-1]      # Last column
actual = newPredictions.iloc[:,-2]         # 2nd last column

plt.scatter(actual, predicted)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Actual Vs Predicted')
plt.savefig("result-scatter-plot.jpg", dpi=300)
plt.show()
```



3.7 Save prediction results to csv

```
[ ] newPredictions.to_csv("NewPredictions.csv")
    # No output
```

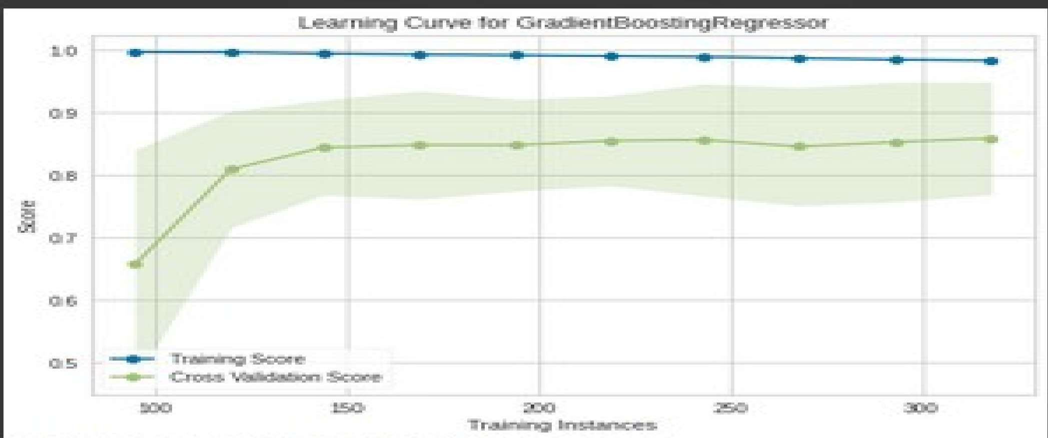
3.8 Plot the Model

Following parameter can be plot for model

- Prediction Error Plot - 'error'
- Learning Curve - 'learning'
- Validation Curve - 'vc'
- Feature Importance - 'feature'
- Model Hyperparameter - 'parameter'

3.8.2 Plot Error (Scatter Plot)

```
[ ] plot_model(gbrModel, plot='learning')
```

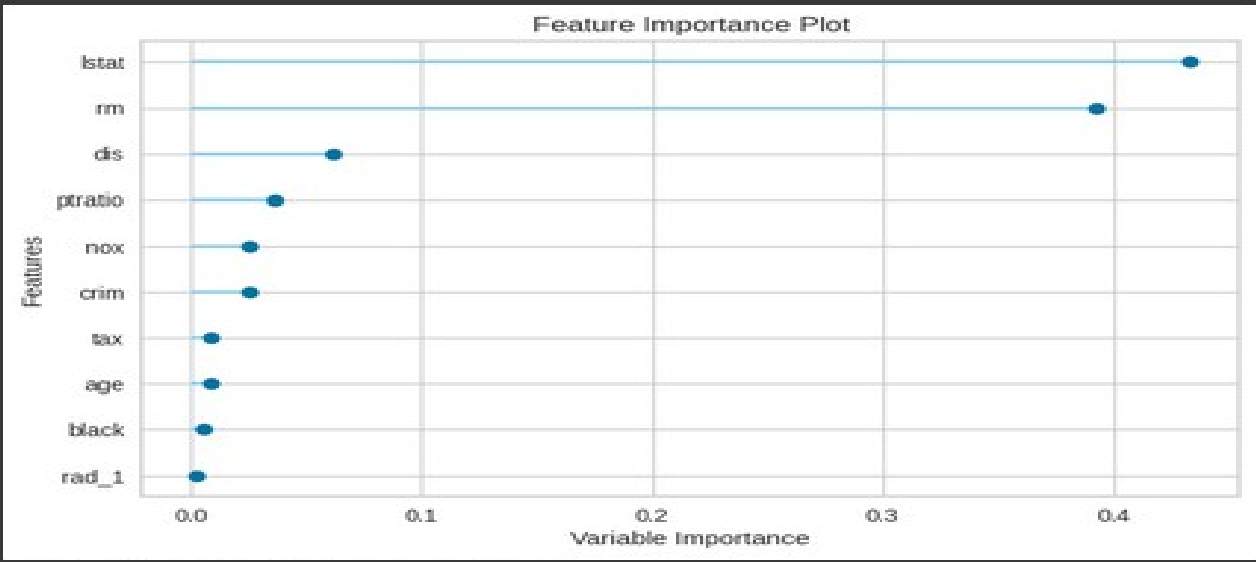



```
INFO:logs:Visual Rendered Successfully
INFO:logs:plot_model() succesfully completed................
```

3.9 Feature Importance

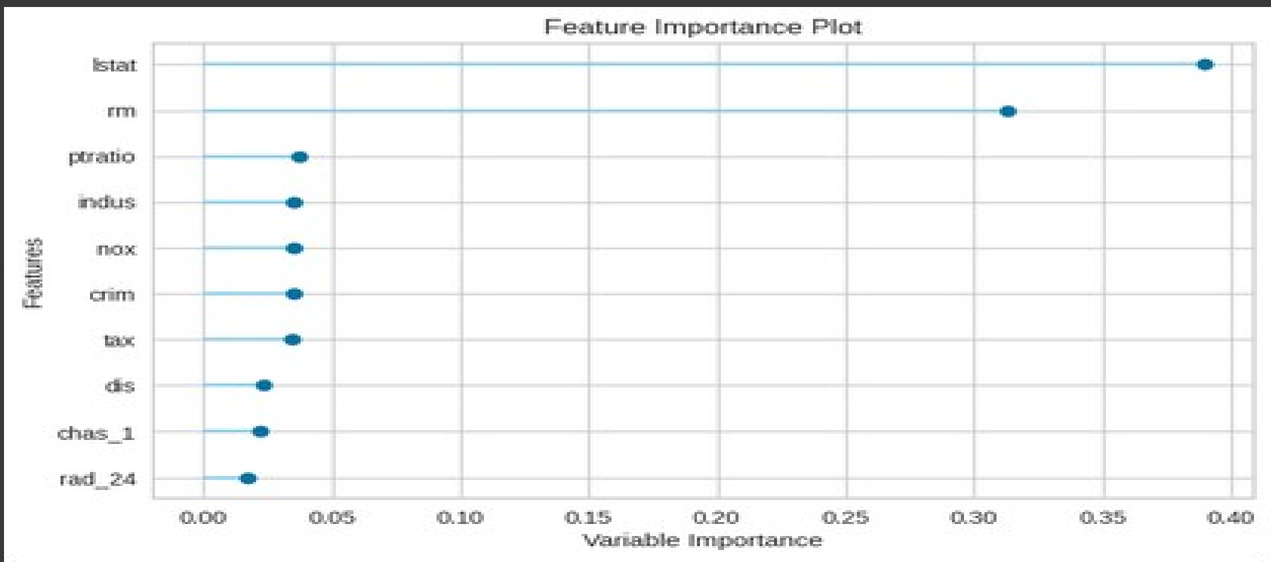
3.9.1 Feature Importance using Random Forest

```
rfModel = create_model('gbr', verbose=False)
plot_model(rfModel, plot='feature')
```



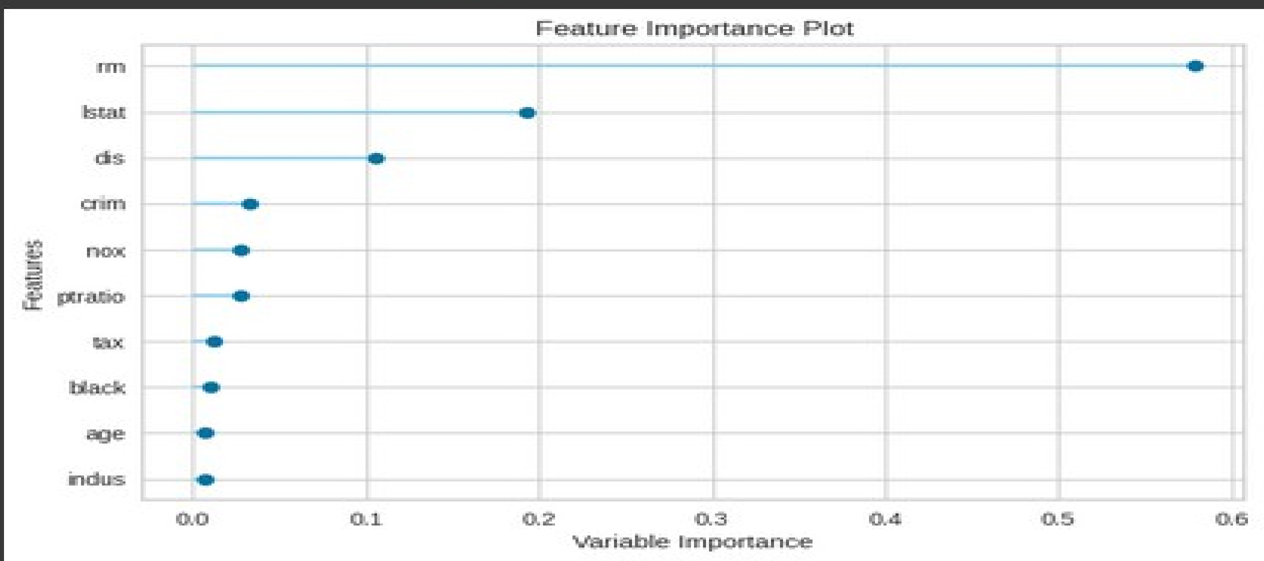
3.9.2 Feature Importance using Extra Trees Regressor

```
etModel = create_model('et', verbose=False)
plot_model(etModel, plot='feature')
```



3.9.3 Feature Importance using Decision Tree

```
dtModel = create_model('dt', verbose=False)
plot_model(dtModel, plot='feature')
```



University Institute of Engineering

Department of Computer Science & Engineering

Evaluation Grid (To be filled by Faculty):

Sr. No.	Parameters	Marks Obtained	Maximum Marks
1.	Worksheet completion including writinglearning objectives/Outcomes. (To be submitted at the end of the day)		10
2.	Post Lab Quiz Result.		5
3.	Student Engagement in Simulation/Demonstration/Performance and Controls/Pre-Lab Questions.		5
	Signature of Faculty (with Date):	**Total Marks Obtained:**	**20**