PRACTICAL 4

AIM: Read the dataset from Car-Dekho to perform multilinear regression. Perform different data analysis and data visualization operations on it. Apply support vector regression and understand the concept of hyperparameter tuning. Compare the performance of the SVR model before and after fine tuning by using the GridSearchCV method.

Code: import
numpy as np
import pandas as pd
import matplotlib.pyplot as plt

df=pd.read_csv("/content/car data.csv")

print(df.shape)
df.head()

Output:

```
print(df.shape)
df.head()
```

```
(301, 9)
```

	Selling_Price	Present_Price	Kms_Driven	Owner	Vehicle_age	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_Individual	Transmission_Manual
0	3.35	5.59	27000	0	10	False	True	False	True
1	4.75	9.54	43000	0	11	True	False	False	True
2	7.25	9.85	6900	0	7	False	True	False	True
3	2.85	4.15	5200	0	13	False	True	False	True
4	4.60	6.87	42450	0	10	True	False	False	True

df.info()

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 301 entries, 0 to 300
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Car_Name       301 non-null    object
 1   Year           301 non-null    int64
 2   Selling_Price  301 non-null    float64
 3   Present_Price  301 non-null    float64
 4   Kms_Driven     301 non-null    int64
 5   Fuel_Type      301 non-null    object
 6   Seller_Type    301 non-null    object
 7   Transmission   301 non-null    object
 8   Owner          301 non-null    int64
dtypes: float64(2), int64(3), object(4)
memory usage: 21.3+ KB
```

Name: DHRUV SHERE
Enrollment No: 23012022021
Batch: 5IT-B-2

df.describe()

```
df.describe()
```

	Year	Selling_Price	Present_Price	Kms_Driven	Owner
count	301.000000	301.000000	301.000000	301.000000	301.000000
mean	2013.627907	4.661296	7.628472	36947.205980	0.043189
std	2.891554	5.082812	8.644115	38886.883882	0.247915
min	2003.000000	0.100000	0.320000	500.000000	0.000000
25%	2012.000000	0.900000	1.200000	15000.000000	0.000000
50%	2014.000000	3.600000	6.400000	32000.000000	0.000000
75%	2016.000000	6.000000	9.900000	48767.000000	0.000000
max	2018.000000	35.000000	92.600000	500000.000000	3.000000

df.isnull().sum()

```
df.isnull().sum()
```

	0
Car_Name	0
Year	0
Selling_Price	0
Present_Price	0
Kms_Driven	0
Fuel_Type	0
Seller_Type	0
Transmission	0
Owner	0

dtype: int64

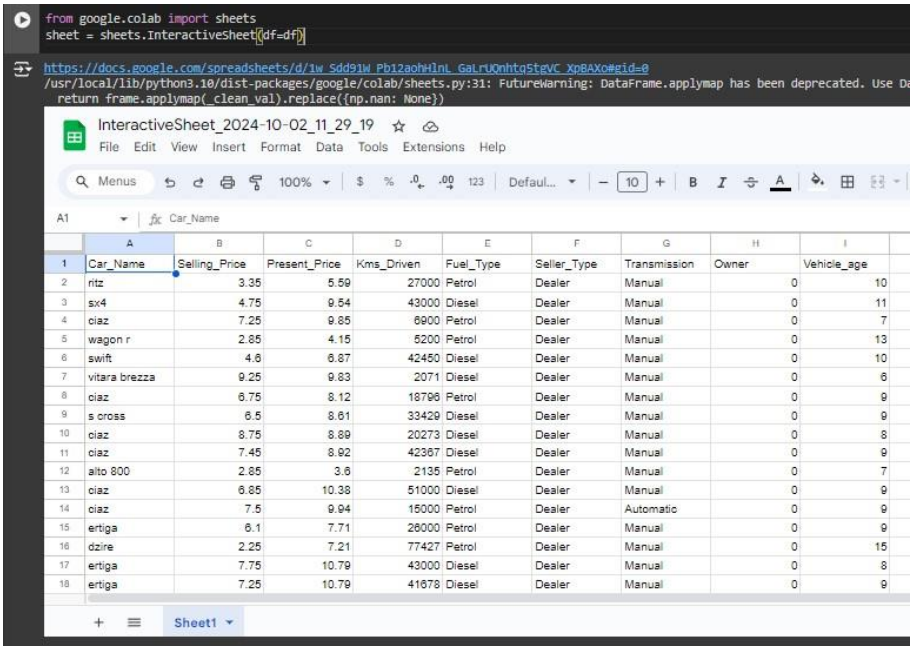
df['Vehicle_age']=2024 - df['Year']
df.drop('Year', axis=1, inplace=True)
df.head()

```
df.head()
```

	Car_Name	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	Vehicle_age
0	ritz	3.35	5.59	27000	Petrol	Dealer	Manual	0	10
1	sx4	4.75	9.54	43000	Diesel	Dealer	Manual	0	11
2	ciaz	7.25	9.85	6900	Petrol	Dealer	Manual	0	7
3	wagon r	2.85	4.15	5200	Petrol	Dealer	Manual	0	13
4	swift	4.60	6.87	42450	Diesel	Dealer	Manual	0	10

```
from google.colab import sheets
sheet = sheets.InteractiveSheet(df=df)
```

Output:

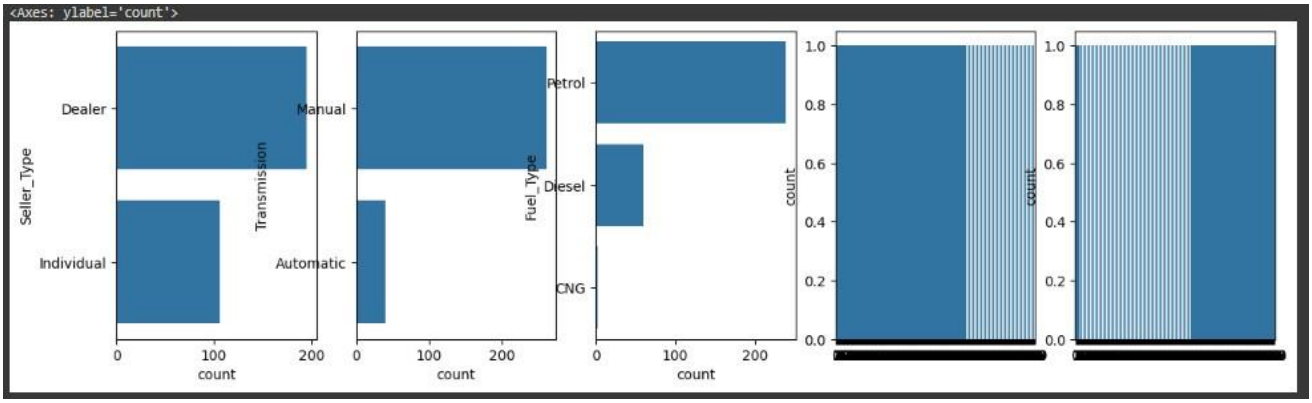


InteractiveSheet_2024-10-02_11_29_19

	Car_Name	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	Vehicle_age
2	ritz	3.35	5.59	27000	Petrol	Dealer	Manual	0	10
3	sx4	4.75	9.54	43000	Diesel	Dealer	Manual	0	11
4	ciaz	7.25	9.85	6900	Petrol	Dealer	Manual	0	7
5	wagon r	2.85	4.15	5200	Petrol	Dealer	Manual	0	13
6	swift	4.6	6.87	42450	Diesel	Dealer	Manual	0	10
7	vitara brezza	9.25	9.83	2071	Diesel	Dealer	Manual	0	6
8	ciaz	6.75	8.12	18796	Petrol	Dealer	Manual	0	9
9	s cross	6.5	8.61	33429	Diesel	Dealer	Manual	0	9
10	ciaz	8.75	8.89	20273	Diesel	Dealer	Manual	0	8
11	ciaz	7.45	8.92	42367	Diesel	Dealer	Manual	0	9
12	alto 800	2.85	3.6	2135	Petrol	Dealer	Manual	0	7
13	ciaz	6.85	10.38	51000	Diesel	Dealer	Manual	0	9
14	ciaz	7.5	9.94	15000	Petrol	Dealer	Automatic	0	9
15	ertiga	6.1	7.71	26000	Petrol	Dealer	Manual	0	9
16	dzire	2.25	7.21	77427	Petrol	Dealer	Manual	0	15
17	ertiga	7.75	10.79	43000	Diesel	Dealer	Manual	0	8
18	ertiga	7.25	10.79	41678	Diesel	Dealer	Manual	0	9

```
import matplotlib.pyplot as plt  import
seaborn as sns
plt.figure(figsize=(15,4))
plt.subplot(1,5,1)
sns.countplot(df['Seller_Type'])
plt.subplot(1,5,2)
sns.countplot(df['Transmission'])
plt.subplot(1,5,3)
sns.countplot(df['Fuel_Type'])
plt.subplot(1,5,4)
sns.countplot(df['Owner'])
plt.subplot(1,5,5)
sns.countplot(df['Vehicle_age'])
```

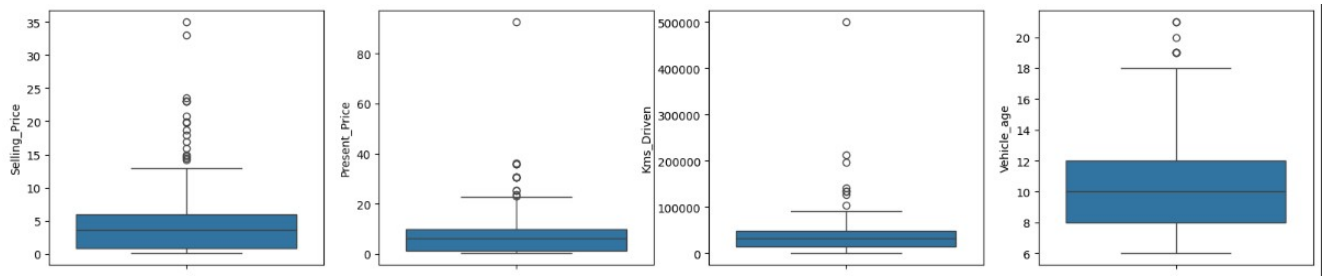
Output:



□ Check the outliers

```
plt.figure(figsize=(25,4))
plt.subplot(1,5,1)
sns.boxplot(df['Selling_Price'])
plt.subplot(1,5,2)
sns.boxplot(df['Present_Price'])
plt.subplot(1,5,3)
sns.boxplot(df['Kms_Driven'])
plt.subplot(1,5,4)
sns.boxplot(df['Vehicle_age'])
```

Output:



df.columns

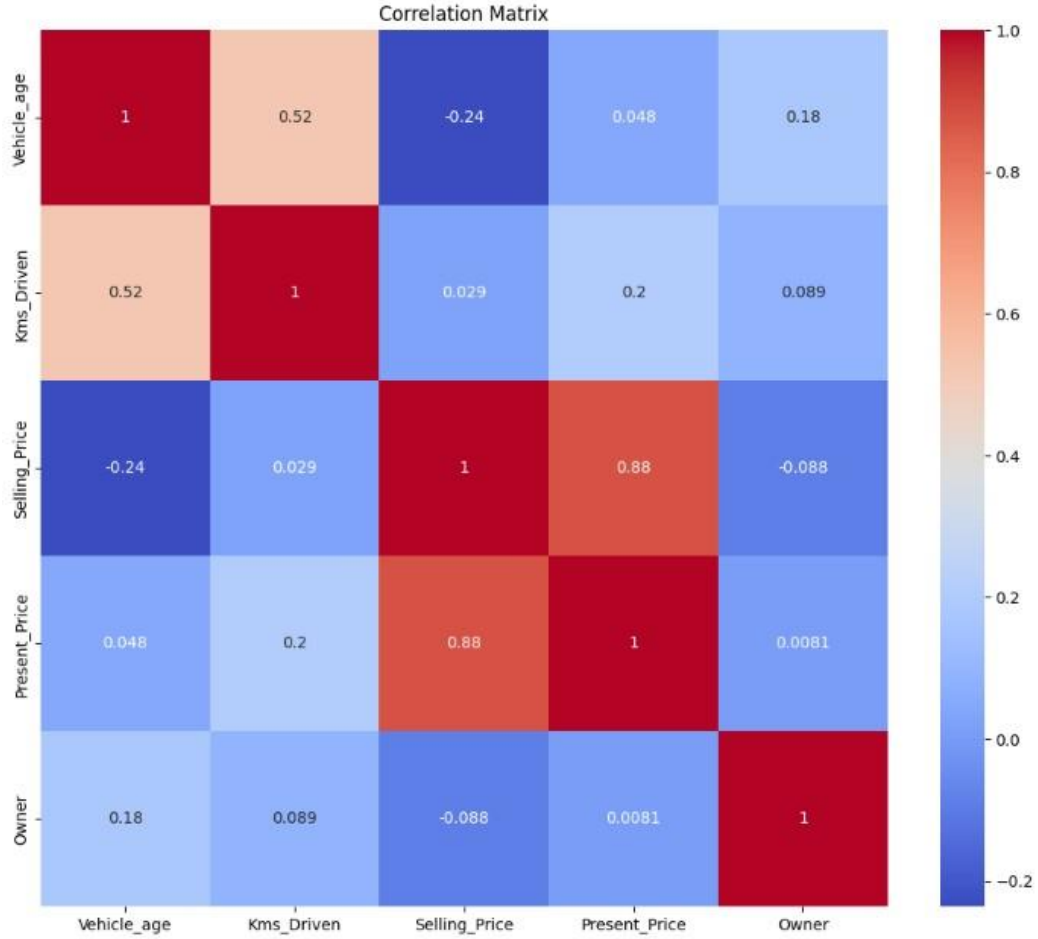
```
df.columns

Index(['Selling_Price', 'Present_Price', 'Kms_Driven', 'Owner', 'Vehicle_age',
       'Fuel_Type_Diesel', 'Fuel_Type_Petrol', 'Seller_Type_Individual',
       'Transmission_Manual'],
      dtype='object')
```

 # Select only numerical columns

numerical_columns = ['Vehicle_age', 'Kms_Driven', 'Selling_Price', 'Present_Price', 'Owner'] numerical_data = df[numerical_columns]

 # Correlation matrix plt.figure(figsize=(12,
10))
sns.heatmap(numerical_data.corr(), annot=True, cmap='coolwarm') plt.title('Correlation Matrix')
plt.show()

Output:



df.drop('Car_Name', axis=1, inplace=True)
print(df.shape)
df.head()

Output:

```
print(df.shape)
df.head()
```

```
(301, 8)
```

	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	Vehicle_age
0	3.35	5.59	27000	Petrol	Dealer	Manual	0	10
1	4.75	9.54	43000	Diesel	Dealer	Manual	0	11
2	7.25	9.85	6900	Petrol	Dealer	Manual	0	7
3	2.85	4.15	5200	Petrol	Dealer	Manual	0	13
4	4.60	6.87	42450	Diesel	Dealer	Manual	0	10

dummies with categorical feature

df=pd.get_dummies(df, drop_first=True)
print(df.shape) df.head()

```
print(df.shape)
df.head()
```

```
(301, 9)
```

	Selling_Price	Present_Price	Kms_Driven	Owner	Vehicle_age	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_Individual	Transmission_Manual
0	3.35	5.59	27000	0	10	False	True	False	True
1	4.75	9.54	43000	0	11	True	False	False	True
2	7.25	9.85	6900	0	7	False	True	False	True
3	2.85	4.15	5200	0	13	False	True	False	True
4	4.60	6.87	42450	0	10	True	False	False	True

```
X=df.iloc[:, 1:].values
y=df.iloc[:,0].values
from sklearn.model_selection import train_test_split
x_train ,x_test, y_train ,y_test=train_test_split(X, y, test_size=0.30, random_state=1) x_train.shape,
x_test.shape, y_train.shape, y_test.shape
```

Output:

((210, 8), (91, 8), (210,), (91,))

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler() x_train_scale=scaler.fit_transform(x_train)
x_test_scale=scaler.fit_transform(x_test)
```

□ Linear Regression

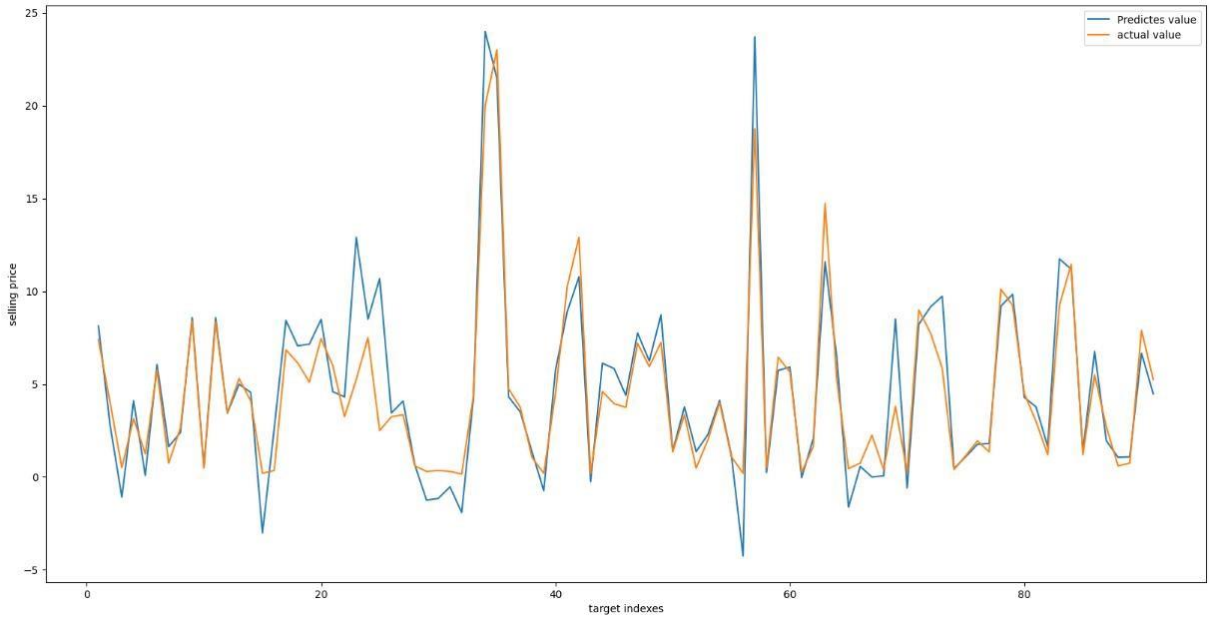
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression() lr.fit(x_train_scale ,y_train)
print('training set score: {}'.format(lr.score(x_train_scale ,y_train))) print("testing
set score :{}".format(lr.score(x_test_scale ,y_test)))
```

Output:

```
training set score: 0.8838482603960781
testing set score :0.8147870024387313
```

```
y_pred=lr.predict(x_test_scale)
plt.figure(figsize=(20,10)) index=range(1,
len(y_pred)+1) plt.plot(index, y_pred,
label='Predictes value') plt.plot(index, y_test,
label='actual value') plt.legend()
plt.xlabel('target indexes') plt.ylabel('selling
price')
plt.show()
```

Output:



```
from sklearn.metrics import r2_score
r2_linear=r2_score(y_test, y_pred)
r2_linear
```

Output:
0.8147870024387313

```
from sklearn.svm import SVR svr=SVR()
svr.fit(x_train_scale ,y_train)
print("training set score : {}".format(svr.score(x_train_scale, y_train)))
print("training set score : {}".format(svr.score(x_test_scale, y_test))) y_pred1=svr.predict(x_test_scale)
```

Output: training set score :
0.6125697083606265 training set score :
0.7455387919116832

```
svr1=SVR()
paras={'C':[0.001, 0.1, 1, 2, 5 ,10, 100],
    'degree':[1,2,3,4,5,6,7],
    'gamma':[0.001, 0.1, 1,2 ,5, 10, 100]}
```

```
from sklearn.model_selection import GridSearchCV svr1=SVR()
paras={'C':[0.001, 0.1, 1, 2, 5 ,10, 100],
    'degree':[1,2,3,4,5,6,7],
    'gamma':[0.001, 0.1, 1,2 ,5, 10, 100]}
```

```
gridsearch=GridSearchCV(estimator=svr,param_grid=paras, cv=10, n_jobs=-1)
gridsearch.fit(x_train_scale,y_train)
```
Output:


```
gridsearch.best_params_
```
Output:
{'C': 100, 'degree': 1, 'gamma': 0.1}

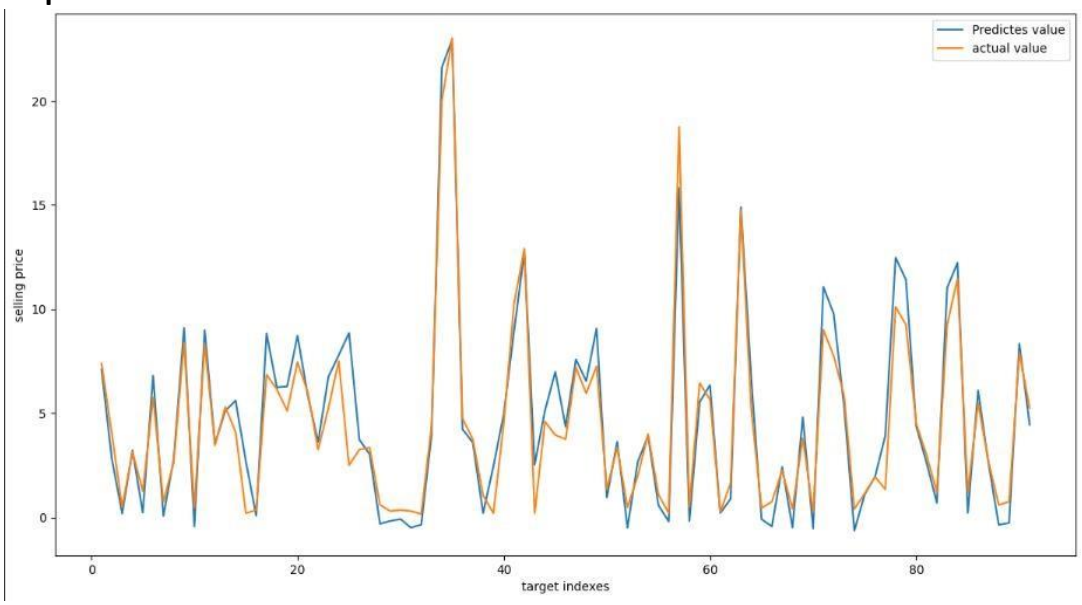
Model with best parameters

```
svr=SVR(C=100, degree=1, gamma=0.1) svr.fit(x_train_scale ,y_train)
print("training set score : {}".format(svr.score(x_train_scale, y_train)))
print("testing set score : {}".format(svr.score(x_test_scale, y_test)))
y_pred2=svr.predict(x_test_scale)
```

Output: training set score :
0.9911796222913005
testing set score : 0.9152461397323125

```
plt.figure(figsize=(15,8)) index=range(1,
len(y_pred2)+1) plt.plot(index, y_pred2,
label='Predictes value') plt.plot(index, y_test,
label='actual value') plt.legend()
plt.xlabel('target indexes') plt.ylabel('selling
price')
plt.show()
```

Output:




```
r2_svm=r2_score(y_test, y_pred2)
r2_svm
```

Output:
0.9152461397323125

□ Random Forest Regressor

```
from sklearn.ensemble import RandomForestRegressor
random=RandomForestRegressor() random.fit(x_train ,y_train)
print("training set score :{}".format(random.score(x_train,y_train)))
print("testing set score : {}".format(random.score(x_test,  y_test)))
```

Output:

```
training set score :0.9876957100426992
testing set score : 0.9044622047655345
```

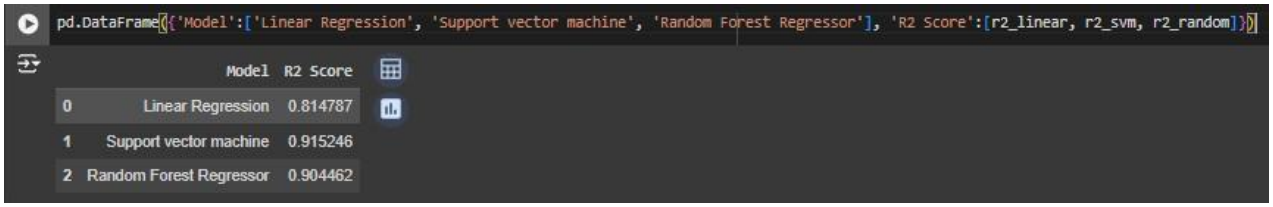
```
y_pred1=random.predict(x_test)
r2_random=r2_score(y_test, y_pred1)
r2_random
```

Output:

```
0.9044622047655345
```

```
pd.DataFrame({'Model':['Linear Regression', 'Support vector machine', 'Random Forest Regressor'], 'R2 Score':[r2_linear, r2_svm, r2_random]})
```

Output:



	Model	R2 Score
0	Linear Regression	0.814787
1	Support vector machine	0.915246
2	Random Forest Regressor	0.904462