

```
In [3]: import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing, svm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
```

```
In [4]: df=pd.read_csv(r"C:\Users\rubin\Downloads\Advertising.csv")
df
```

```
Out[4]:
```

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	12.0
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9
...	...	...	...	...
195	38.2	3.7	13.8	7.6
196	94.2	4.9	8.1	14.0
197	177.0	9.3	6.4	14.8
198	283.6	42.0	66.2	25.5
199	232.1	8.6	8.7	18.4

200 rows × 4 columns

In [5]: `df.head(15)`

Out[5]:

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	12.0
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9
5	8.7	48.9	75.0	7.2
6	57.5	32.8	23.5	11.8
7	120.2	19.6	11.6	13.2
8	8.6	2.1	1.0	4.8
9	199.8	2.6	21.2	15.6
10	66.1	5.8	24.2	12.6
11	214.7	24.0	4.0	17.4
12	23.8	35.1	65.9	9.2
13	97.5	7.6	7.2	13.7
14	204.1	32.9	46.0	19.0

In [6]: `df.tail()`

Out[6]:

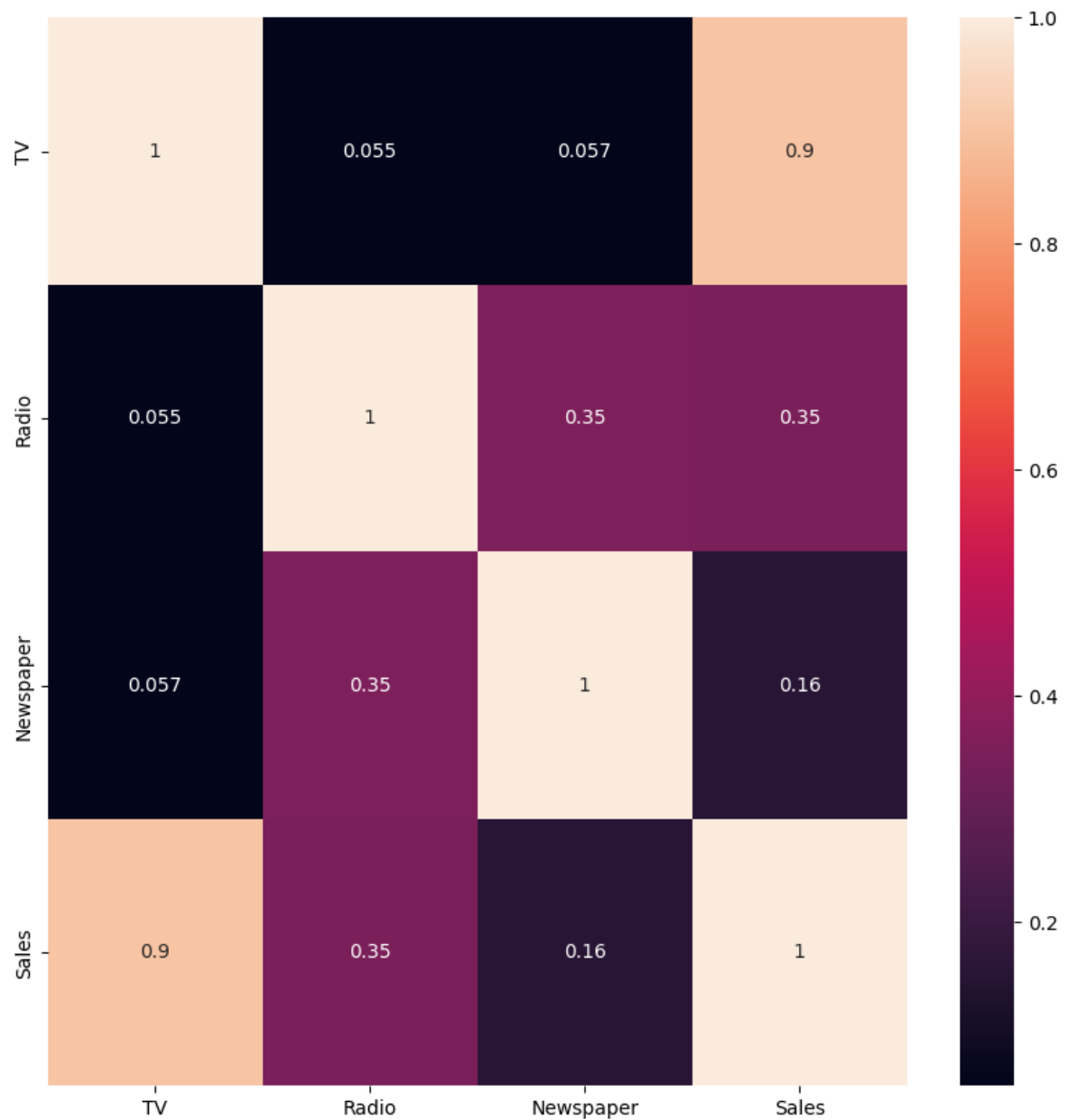
	TV	Radio	Newspaper	Sales
195	38.2	3.7	13.8	7.6
196	94.2	4.9	8.1	14.0
197	177.0	9.3	6.4	14.8
198	283.6	42.0	66.2	25.5
199	232.1	8.6	8.7	18.4

In [7]: `df.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
#   Column      Non-Null Count  Dtype
---  -
0    TV          200 non-null    float64
1    Radio       200 non-null    float64
2    Newspaper   200 non-null    float64
3    Sales       200 non-null    float64
dtypes: float64(4)
memory usage: 6.4 KB
```

```
In [8]: plt.figure(figsize=(10,10))  
sns.heatmap(df.corr(),annot=True)
```

Out[8]: <Axes: >



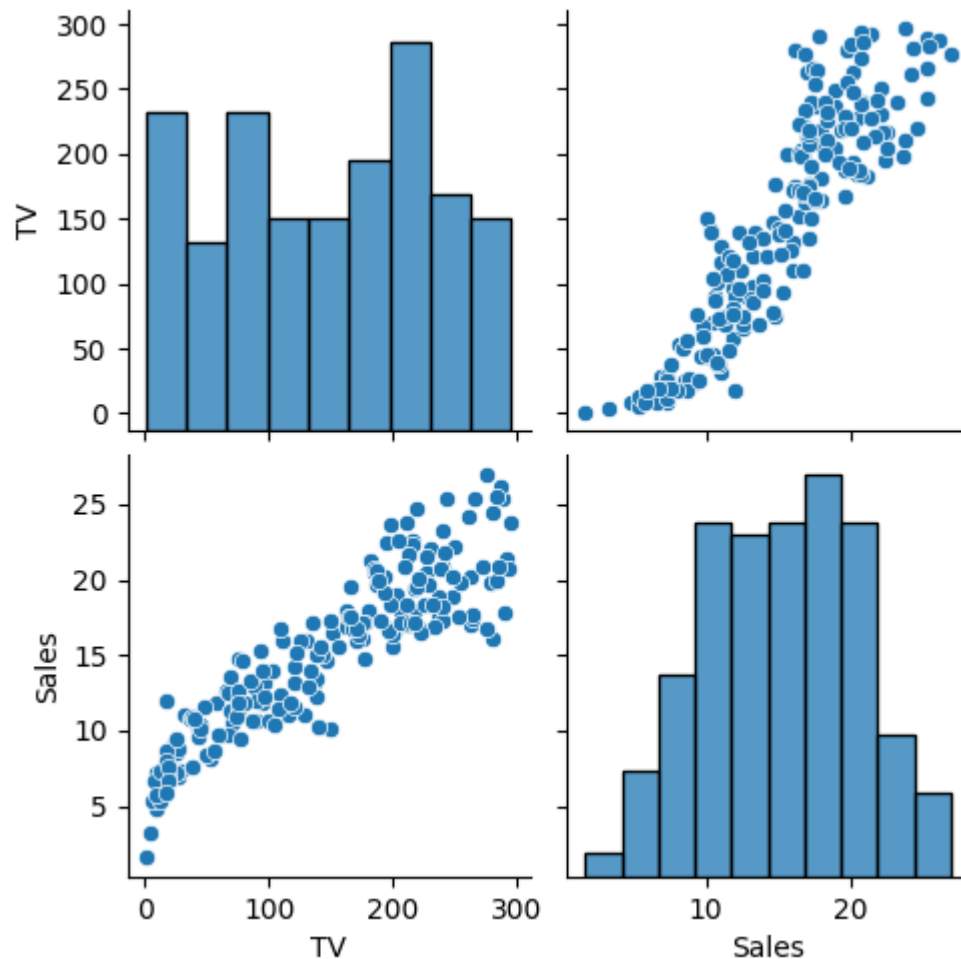
## Ridge and Lasso Regression

```
In [9]: from sklearn.linear_model import Lasso,Ridge
```

```
In [10]: df.drop(columns=["Radio", "Newspaper"], inplace=True)
sns.pairplot(df)
df.sales=np.log(df.Sales)
```

C:\Users\rubin\AppData\Local\Temp\ipykernel\_22748\1465564857.py:3: UserWarning: Pandas doesn't allow columns to be created via a new attribute name - see <https://pandas.pydata.org/pandas-docs/stable/indexing.html#attribute-access> (<https://pandas.pydata.org/pandas-docs/stable/indexing.html#attribute-access>)

```
df.sales=np.log(df.Sales)
```



```
In [11]: features = df.columns[0:2]
target = df.columns[-1]
#X and y values
X = df[features].values
y = df[target].values
#split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
print("The dimension of X_train is {}".format(X_train.shape))
print("The dimension of X_test is {}".format(X_test.shape))
#Scale features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

The dimension of X\_train is (140, 2)

The dimension of X\_test is (60, 2)

```
In [12]: lr = LinearRegression()
#Fit model
lr.fit(X_train, y_train)
#predict
#prediction = lr.predict(X_test)
#actual
actual = y_test
train_score_lr = lr.score(X_train, y_train)
test_score_lr = lr.score(X_test, y_test)
print("\nLinear Regression Model:\n")
print("The train score for lr model is {}".format(train_score_lr))
print("The test score for lr model is {}".format(test_score_lr))
```

Linear Regression Model:

The train score for lr model is 1.0

The test score for lr model is 1.0

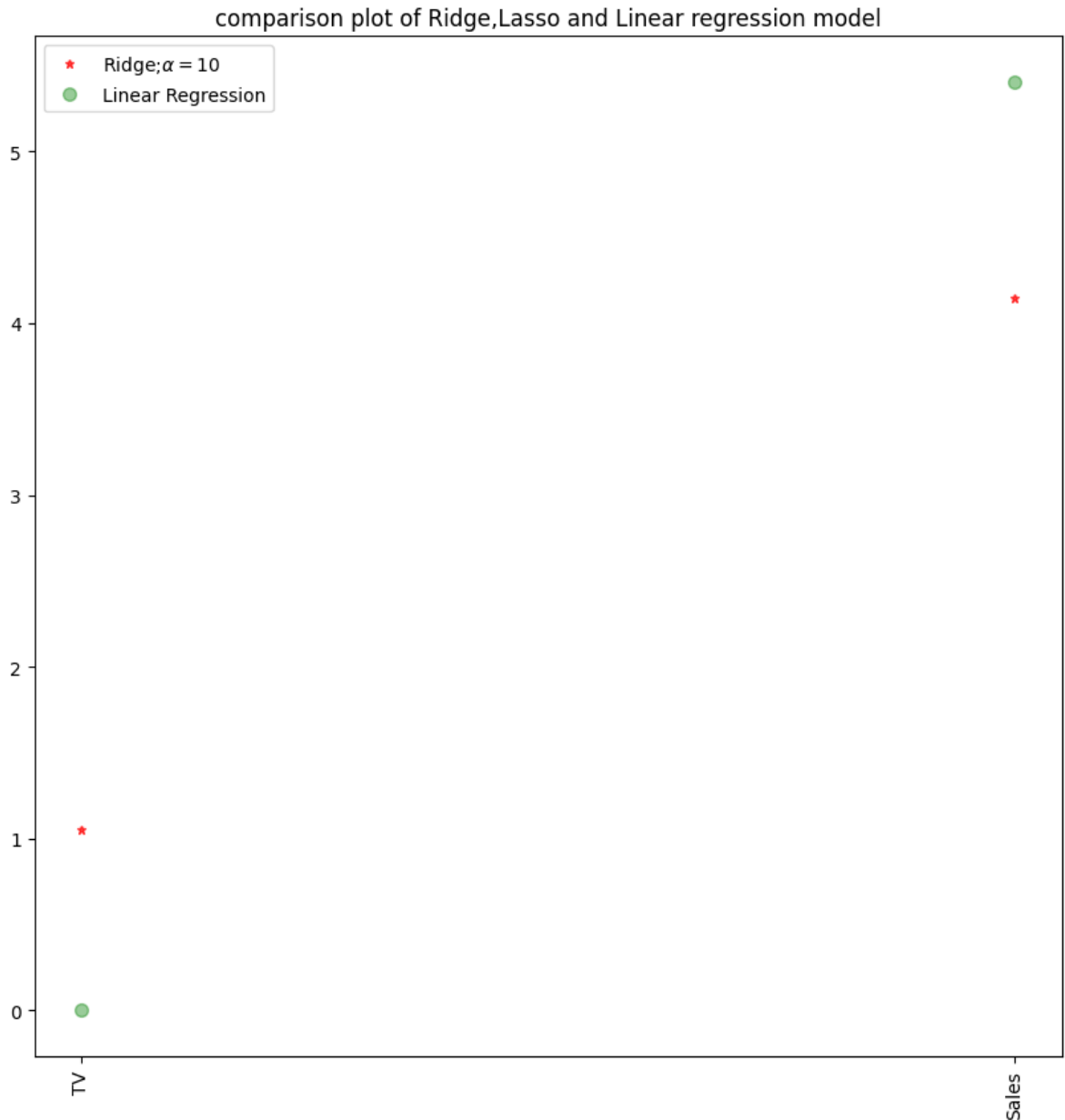
```
In [13]: #Ridge Regression Model
ridgeReg = Ridge(alpha=10)
ridgeReg.fit(X_train, y_train)
#train and test score for ridge regression
train_score_ridge = ridgeReg.score(X_train, y_train)
test_score_ridge = ridgeReg.score(X_test, y_test)
print("\nRidge Model:\n")
print("The train score for ridge model is {}".format(train_score_ridge))
print("The test score for ridge model is {}".format(test_score_ridge))
```

Ridge Model:

The train score for ridge model is 0.9900167746680466

The test score for ridge model is 0.9888279083610404

```
In [14]: plt.figure(figsize=(10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=6,color='red')
#plt.plot(rr100.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue')
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green')
plt.xticks(rotation=90)
plt.legend()
plt.title("comparison plot of Ridge,Lasso and Linear regression model")
plt.show()
```



```
In [15]: print("\nLasso Model: \n")
lasso = Lasso(alpha = 10)
lasso.fit(X_train,y_train)
train_score_ls =lasso.score(X_train,y_train)
test_score_ls =lasso.score(X_test,y_test)
print("The train score for ls model is {}".format(train_score_ls))
print("The test score for ls model is {}".format(test_score_ls))
```

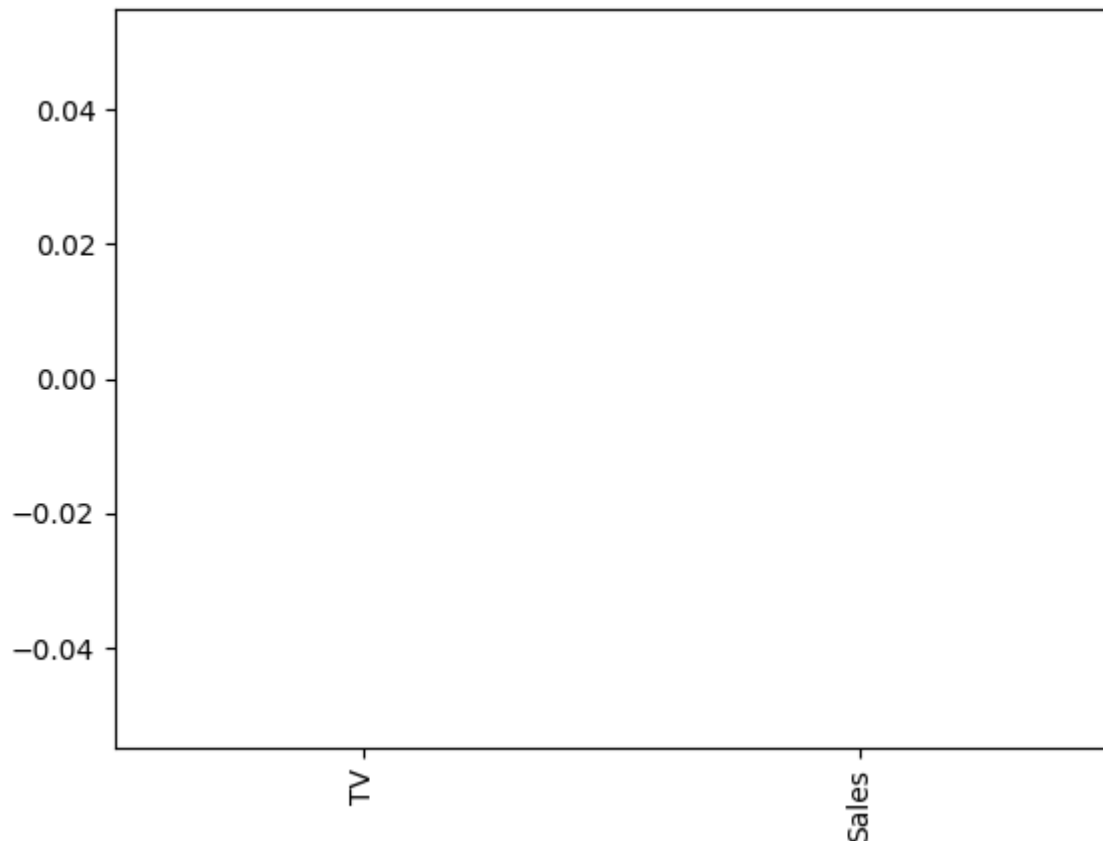
Lasso Model:

The train score for ls model is 0.0

The test score for ls model is -0.0064111102763571015

```
In [16]: pd.Series(lasso.coef_, features).sort_values(ascending = True).plot(kind = "bar")
```

Out[16]: <Axes: >

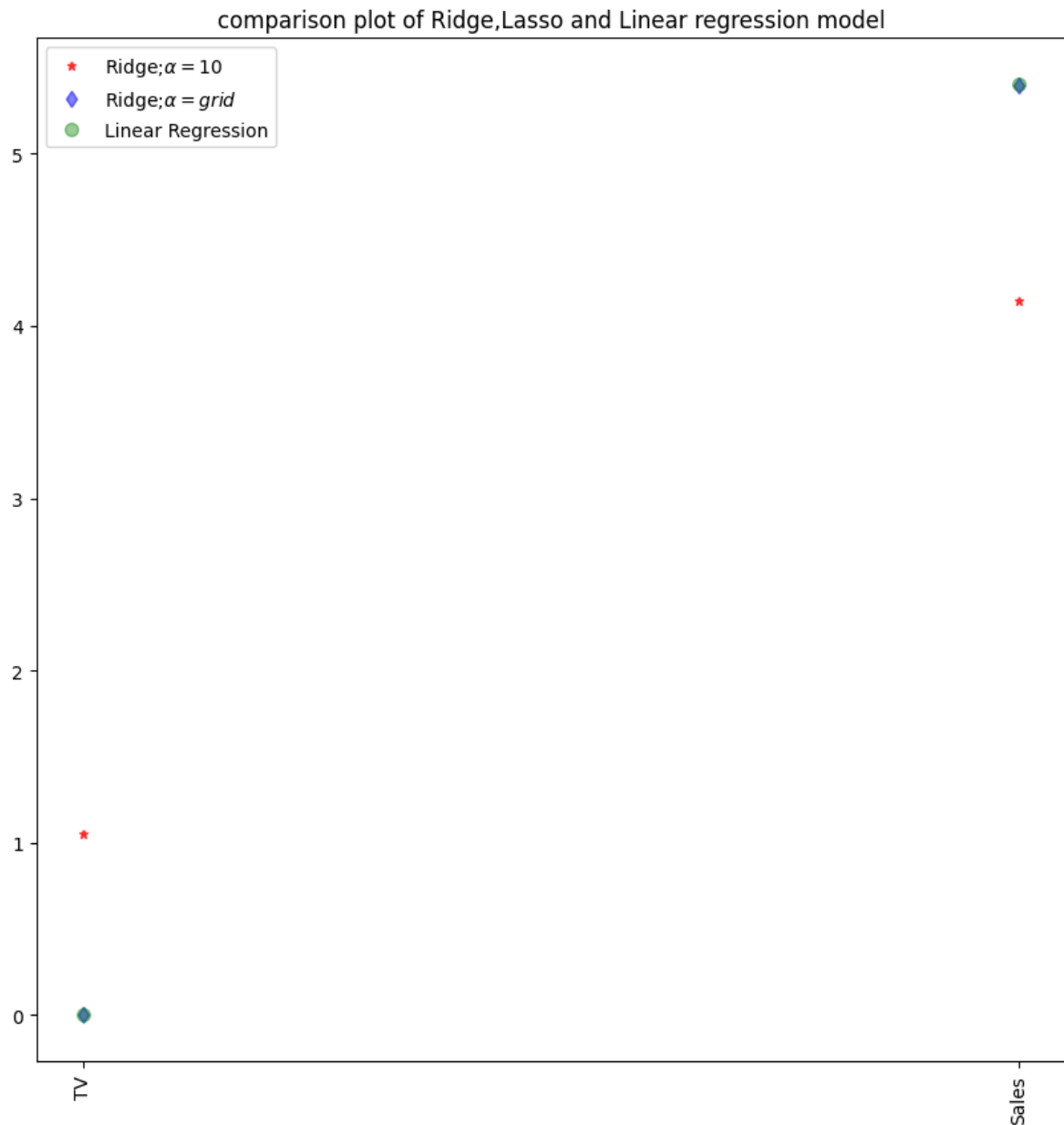


```
In [17]: from sklearn.linear_model import LassoCV
#Lasso Cross validation
lasso_cv = LassoCV(alphas = [0.0001, 0.001,0.01, 0.1, 1, 10], random_state=0).
#score
print(lasso_cv.score(X_train, y_train))
print(lasso_cv.score(X_test, y_test))
```

0.9999999677147366

0.9999999641980227

```
In [18]: plt.figure(figsize=(10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=6,color='red')
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue')
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green')
plt.xticks(rotation=90)
plt.legend()
plt.title("comparison plot of Ridge,Lasso and Linear regression model")
plt.show()
```





```
In [19]: from sklearn.linear_model import RidgeCV
#Ridge Cross validation
ridge_cv = RidgeCV(alphas = [0.0001, 0.001,0.01, 0.1, 1, 10]).fit(X_train, y_train)
#score
print("The train score for ridge model is {}".format(ridge_cv.score(X_train, y_train)))
print("The train score for ridge model is {}".format(ridge_cv.score(X_test, y_test)))
```

The train score for ridge model is 0.99999999997204  
The train score for ridge model is 0.999999999968613

## Elastic Net Regression

```
In [23]: from sklearn.linear_model import ElasticNet
regr=ElasticNet()
regr.fit(X,y)
print(regr.coef_)
print(regr.intercept_)
```

[0.00938134 0.82969623]  
1.197325903826

```
In [25]: y_pred_elastic=regr.predict(X_train)
```

```
In [27]: mean_squared_error=np.mean((y_pred_elastic-y_train)**2)
print("Mean Squared Error on test set",mean_squared_error)
```

Mean Squared Error on test set 218.26629572962375

```
In [ ]:
```