

```
In [ ]: 1 df500.fillna(method = 'ffill', inplace = True)
2 X = np.array(df500['AAH']).reshape(-1,1)
3 y = np.array(df500['prc']).reshape(-1,1)
4 df500.dropna(inplace = True)
5 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25)
6 regr = LinearRegression()
7 regr.fit(X_train, y_train)
8 print("Regression: ",regr.score(X_test, y_test))
9 y_pred = regr.predict(X_test)
10 plt.scatter(X_test, y_test, color = 'b')
11 plt.plot(X_test, y_pred, color = 'k')
12 plt.show
```

```
In [ ]: 1 from sklearn.linear_model import LinearRegression
2 from sklearn.metrics import r2_score
3 # Train the model
4 model = LinearRegression()
5 model.fit(X_train, y_train)
6 y_pred=model.predict(X_test)
7 r2=r2_score(y_test,y_pred)
8 print("R2 score: ",r2)
9 # Evaluate the model on the test set
```

Conclusion

Dataset we have taken is poor for linear model but with the smaller data works well with linear model.

Ridge, RidgeCV & Lasso

```
In [ ]: 1 # for Fiat Vehicles dataset
```

```
In [ ]: 1 features = ['ID','engine_power','age_in_days','km','previous_owners',  
2           'lat','lon']
```

```
In [ ]: 1 target=['price']
```

```
In [ ]: 1 from sklearn.linear_model import Ridge,RidgeCV,Lasso  
2 from sklearn.linear_model import LinearRegression  
3 from sklearn.preprocessing import StandardScaler  
4 from sklearn.model_selection import train_test_split
```

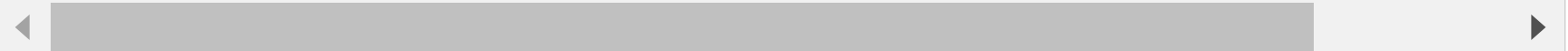
```
In [ ]: 1 ridgeReg = Ridge(alpha=10)  
2 ridgeReg.fit(x_train,y_train)  
3 train_score_ridge = ridgeReg.score(x_train,y_train)  
4 test_score_ridge = ridgeReg.score(x_test,y_test)  
5  
6 print('\nRidge model\n')  
7 print('Train score for ridge model is {}'.format(train_score_ridge))  
8 print('Test score for ridge model is {}'.format(test_score_ridge))
```

```
In [ ]: 1 plt.figure(figsize=(5,5))  
2 plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=7,color='Red',label=r'Ridge;$\  
3 plt.plot(features,lm.coef_,alpha=0.4,linestyle='none',color='green',marker='d',markersize=6,label='LinearRegress  
4 plt.xticks(rotation=90)  
5 plt.legend()  
6 plt.show()  
7
```




```
In [ ]: 1 lassoReg=Lasso(alpha=10)
2 lassoReg.fit(x_train,y_train)
3 train_score_lasso=lassoReg.score(x_train,y_train)
4 test_score_lasso=lassoReg.score(x_test,y_test)
5
6 print('\nLasso Model\n')
7 print('Train score for lasso model is {}'.format(train_score_lasso))
8 print('Test score for lasso model is {}'.format(test_score_lasso))
```

```
In [ ]: 1 plt.figure(figsize=(5,5))
2 plt.plot(features,ridgeReg.coef_,alpha=0.5,linestyle='none',marker='*',markersize=6,color='red',label=r'lasso:al
3 plt.plot(features,lm.coef_,alpha=0.6,linestyle='none',marker='d',markersize=7,color='k',label='LinearRegression'
4 plt.xticks(rotation=90)
5 plt.legend()
6 plt.show()
```



```
In [ ]: 1 # comaprison between Ridge,Lasso and ridgeCV
2 plt.figure(figsize=(5,5))
3
4 plt.plot(features,ridgeReg.coef_,alpha=0.4,linestyle='none',marker='*',markersize=7,color='red',label=r'Ridge;\$
5 plt.plot(features,lassoReg.coef_,alpha=0.5,linestyle='none',marker='o',markersize=6,color='green',label=r'lasso;
6 plt.plot(features,lm.coef_,alpha=0.5,linestyle='none',marker='d',markersize=7,color='yellow',label='LinearRegres
7
8 plt.xticks(rotation=90)
9 plt.title('Comparison between Rudge,Lasso and RidgeCV')
10 plt.legend()
11 plt.show()
```



```
In [ ]: 1 # Linear CV model using Ridge
2 from sklearn.linear_model import RidgeCV
3 ridge_CV=RidgeCV(alphas=[0.1,0.4,1.1]).fit(x_train,y_train)
4 print('The Train score for ridge model is {}'.format(ridge_CV.score(x_train,y_train)))
5 print('The Test score for ridge model is {}'.format(ridge_CV.score(x_test,y_test)))
```

```
In [ ]: 1 # Linear CV model using Lasso
2 from sklearn.linear_model import LassoCV
3 lasso_CV=LassoCV(alphas=[1,10,20]).fit(x_train,y_train)
4 print("The train score for lasso model is {}".format(lasso_CV.score(x_train,y_train)))
5 print("The test score for lasso model is {}".format(lasso_CV.score(x_test,y_test)))
```

```
In [ ]: 1 # Elastic Net
2 from sklearn.linear_model import ElasticNet
3 regr = ElasticNet()
4 regr.fit(x,y)
5 print(regr.coef_)
6 print(regr.intercept_)
7 y_pred_elastic = regr.predict(X_train)
8 mean_squared_error = np.mean((y_pred_elastic-y_train)**2)
9 print('Mean squared error on test set',mean_squared_error)
```

```
In [ ]: 1
```

```
In [ ]: 1
```