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

In [ ]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
# Train the model
model = LinearRegression()
model.fit(X_train, y_train)
y_pred=model.predict(X_test)
r2=r2_score(y_test,y_pred)
print("R2 score: ",r2)
# 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 [ ]:
```
# for Fiat Vehicles dataset
```

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

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

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

In [ ]:
```
ridgeReg = Ridge(alpha=10)
ridgeReg.fit(x_train,y_train)
train_score_ridge = ridgeReg.score(x_train,y_train)
test_score_ridge = ridgeReg.score(x_test,y_test)

print('\nRidge model\n')
print('Train score for ridge model is {}'.format(train_score_ridge))
print('Test score for ridge model is {}'.format(test_score_ridge))
```

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

```

In [ ]:
```
lassoReg=Lasso(alpha=10)
lassoReg.fit(x_train,y_train)
train_score_lasso=lassoReg.score(x_train,y_train)
test_score_lasso=lassoReg.score(x_test,y_test)

print('\nLasso Model\n')
print('Train score for lasso model is {}'.format(train_score_lasso))
print('Test score for lasso model is {}'.format(test_score_lasso))
```

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

In [ ]:
```
# comaprison between Ridge,Lasso and ridgeCV
plt.figure(figsize=(5,5))

plt.plot(features,ridgeReg.coef_,alpha=0.4,linestyle='none',marker='*',markersize=7,color='red',label=r'Ridge;\$
plt.plot(features,lassoReg.coef_,alpha=0.5,linestyle='none',marker='o',markersize=6,color='green',label=r'lasso;
plt.plot(features,lm.coef_,alpha=0.5,linestyle='none',marker='d',markersize=7,color='yellow',label='LinearRegres

plt.xticks(rotation=90)
plt.title('Comparison between Rudge,Lasso and RidgeCV')
plt.legend()
plt.show()
```

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

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

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

In [ ]:
```

```

In [ ]:
```

```