* **Build a model which predicts price based on different features given in cars.csv dataset. And then perform Regularization with necessary hyperparameters tuning for a better performing model.**

**# EDA and Preprocessing**

* import pandas as pd
* import numpy as np
* import matplotlib.pyplot as plt
* import seaborn as sns
* import warnings
* warnings.filterwarnings("ignore")
* data = pd.read\_csv("cars.csv")
* data
* data.describe()
* data.info()
* data.dtypes
* data["normalized-losses"].value\_counts()
* data["normalized-losses"].replace("?",np.nan, inplace = True)
* data["normalized-losses"].isnull().sum()
* data["normalized-losses"].value\_counts()
* data["normalized-losses"] = data["normalized-losses"].astype(float)
* data.dtypes
* data\_mean = data["normalized-losses"].mean()
* data["normalized-losses"].replace(np.nan, data\_mean, inplace = True)
* data["normalized-losses"].isnull().sum()
* data
* data["horsepower"].value\_counts()
* data["horsepower"].replace("?", np.nan, inplace = True)
* data = data.dropna()
* data.dtypes
* data["horsepower"] = data["horsepower"].astype(float)
* data.dtypes
* plt.figure(figsize = (10,10))
* sns.boxplot(data = data , x = "make", y = "price")
* plt.xticks(rotation = 90)
* plt.grid()
* plt.show()
* data[(data["make"] == "dodge") & (data["price"] >= 12000)]
* data.drop(29, inplace = True)
* data[(data["make"] == "honda") & (data["price"] >= 12000)]
* data.drop(41, inplace = True)
* data[(data["make"] == "isuzu") & (data["price"] >= 23000)]
* data.drop(45, inplace = True)
* data[(data["make"] == "mitsubishi") & (data["price"] >= 14000)]
* data.drop(index = [83,84], inplace = True)
* data[(data["make"] == "plymouth") & (data["price"] >= 12000)]
* data.drop(124, inplace = True)
* data[(data["make"] == "toyota") & (data["price"] >= 15000)]
* data.drop(index = [172, 178, 179, 180, 181], inplace = True)
* data\_cat = data.select\_dtypes(["object"])
* data\_cat
* data\_num = data.select\_dtypes(["int","float"])
* data\_num

# # Encoding (Label Encoding)

* from sklearn.preprocessing import LabelEncoder
* le = LabelEncoder()
* for col in data\_cat:
* data\_cat[col] = le.fit\_transform(data\_cat[col])
* data\_cat
* data\_new = pd.concat([data\_cat, data\_num], axis = 1)
* data\_new

# # Seperation of x and y.

# x = data\_new.iloc[:,:-1]

# x

# y = data\_new.iloc[:,-1]

# y

# # Model Preparation

# from sklearn.model\_selection import train\_test\_split

# xtrain, xtest, ytrain, ytest = train\_test\_split(x,y, test\_size = 0.3, random\_state = 1)

# from sklearn.linear\_model import LinearRegression

# lr = LinearRegression()

# lr.fit(xtrain, ytrain)

# ypred = lr.predict(xtest)

# ypred

# train = lr.score(xtrain, ytrain)

# test = lr.score(xtest, ytest)

# print(f"Training Accuracy is {train}")

# print(f"Testing Accuracy is {test}")

# # Regularization

# from sklearn.linear\_model import Ridge, Lasso

# # L2 Regularization (RIDGE = Regularization Regression)

# for i in range(1,5):

# l2 = Ridge(alpha = i)

# l2.fit(xtrain, ytrain)

# test = l2.score(xtest, ytest)

# print(f"For Value of Lambda {i} Testing Accuracy is {test}")

# l2 = Ridge(alpha = 1)

# l2.fit(xtrain, ytrain)

# train = l2.score(xtrain, ytrain)

# test = l2.score(xtest, ytest)

# print(f"Training Accuracy is {train}")

# print(f"Testing Accuracy is {test}")

# # Final Prediction

# ypred = l2.predict(xtest)

# ypred # Price Prediction becomes precise after hypertunning Lambda parameter through Ridge Regularization method.

# pred = l2.predict(xtest)[0]

# print(f"The price prediction for given data is {pred}")