```
In [ ]:
          import numpy as np
          import pandas as pd
          import matplotlib.pyplot as plt
In [ ]:
          columns = np.array(["symboling","normailsed-losses","make","fuel-type","aspiration","nu
          print(columns.shape)
         (26,)
In [ ]:
          data = pd.read_csv("imports-85.csv")
          data.columns = columns
          data.head()
Out[]:
                                                              num-
                        normailsed-
                                                                         body-
                                                                                 drive-
                                                                                         engine-
                                                                                                  wheel-
                                             fuel-
             symboling
                                      make
                                                   aspiration
                                                                of-
                             losses
                                                                                wheels locations
                                             type
                                                                          style
                                                                                                    base
                                                              doors
                                       alfa-
         0
                     3
                                  ?
                                                                                                     88.6
                                                                     convertible
                                                                                            front
                                                         std
                                                                two
                                                                                   rwd
                                              gas
                                     romero
                                       alfa-
         1
                     1
                                                         std
                                                                two
                                                                      hatchback
                                                                                   rwd
                                                                                            front
                                                                                                     94.5
                                              gas
                                     romero
         2
                     2
                               164
                                                                                                     99.8
                                       audi
                                                         std
                                                               four
                                                                         sedan
                                                                                   fwd
                                                                                            front
                                              gas
                                                                         sedan
         3
                     2
                               164
                                       audi
                                              gas
                                                         std
                                                               four
                                                                                   4wd
                                                                                            front
                                                                                                     99.4
                     2
                                 ?
                                       audi
                                                                         sedan
                                                                                   fwd
                                                                                            front
                                                                                                     99.8
                                              gas
                                                         std
                                                                two
        5 rows × 26 columns
In [ ]:
          data clean = data.replace(to replace='?',value = np.nan)
          data clean.head()
```

Out[]:		symboling	normailsed- losses	make	fuel- type	aspiration	num- of- doors	body- style	drive- wheels	engine- locations	wheel- base	•••
	0	3	NaN	alfa- romero	gas	std	two	convertible	rwd	front	88.6	•••
	1	1	NaN	alfa- romero	gas	std	two	hatchback	rwd	front	94.5	•••
	2	2	164	audi	gas	std	four	sedan	fwd	front	99.8	
	3	2	164	audi	gas	std	four	sedan	4wd	front	99.4	
	4	2	NaN	audi	gas	std	two	sedan	fwd	front	99.8	

5 rows × 26 columns

```
data clean = data clean.fillna(value = data clean.median)
In [ ]:
         data clean.info()
        <class 'pandas.core.frame.DataFrame'>
        RangeIndex: 204 entries, 0 to 203
        Data columns (total 26 columns):
             Column
                                Non-Null Count Dtype
         #
        ---
             -----
                                -----
                                               ----
         0
                                204 non-null
                                               int64
             symboling
         1
             normailsed-losses 204 non-null
                                               object
         2
                                204 non-null
                                               object
         3
             fuel-type
                                204 non-null
                                               object
         4
             aspiration
                                204 non-null
                                               object
         5
             num-of-doors
                                204 non-null
                                               object
         6
             body-style
                                204 non-null
                                               object
         7
             drive-wheels
                                204 non-null
                                               object
         8
             engine-locations
                                204 non-null
                                               object
         9
             wheel-base
                                               float64
                                204 non-null
         10 length
                                204 non-null
                                                float64
         11 width
                                204 non-null
                                               float64
         12 height
                                204 non-null
                                               float64
         13 curb-weight
                                204 non-null
                                               int64
         14 engine-type
                                204 non-null
                                               object
         15 num-of-cylinders
                                204 non-null
                                               object
         16 engine-size
                                204 non-null
                                               int64
         17 fuel-system
                                204 non-null
                                               object
         18 bore
                                204 non-null
                                               object
         19 stroke
                                204 non-null
                                               object
                                               float64
         20 compression-ratio 204 non-null
         21 horsepower
                                204 non-null
                                               object
         22 peak-rpm
                                204 non-null
                                               object
                                                int64
         23 city-mpg
                                204 non-null
         24 highway-mpg
                                204 non-null
                                                int64
         25 price
                                204 non-null
                                                object
        dtypes: float64(5), int64(5), object(16)
        memory usage: 41.6+ KB
In [ ]:
         from sklearn.preprocessing import LabelEncoder
         label = LabelEncoder()
         col = ['make', 'fuel-type', 'aspiration', 'num-of-doors', 'body-style', 'drive-wheels', 'engi
         for i in col:
             data clean[i] = label.fit transform(data clean[i].astype(str))
In [ ]:
         data_clean.info()
        <class 'pandas.core.frame.DataFrame'>
        RangeIndex: 204 entries, 0 to 203
        Data columns (total 26 columns):
         #
             Column
                                Non-Null Count
                                               Dtype
                                -----
        ---
                                               ----
         0
             symboling
                                204 non-null
                                               int64
         1
             normailsed-losses 204 non-null
                                               int32
         2
             make
                                204 non-null
                                               int32
         3
             fuel-type
                                204 non-null
                                             int32
         4
             aspiration
                                204 non-null
                                               int32
         5
             num-of-doors
                                204 non-null
                                               int32
         6
             body-style
                                204 non-null
                                               int32
         7
             drive-wheels
                                204 non-null
                                                int32
             engine-locations
                                204 non-null
                                                int32
```

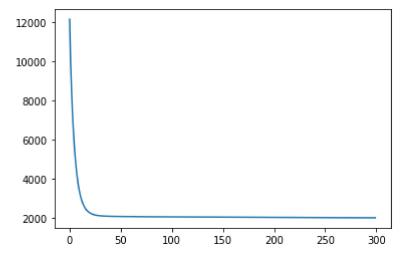
```
wheel-base
                              204 non-null
                                             float64
         10 length
                              204 non-null
                                             float64
         11 width
                              204 non-null
                                             float64
         12 height
                              204 non-null
                                             float64
         13 curb-weight
                              204 non-null
                                             int64
         14 engine-type
                              204 non-null
                                             int32
         15 num-of-cylinders 204 non-null
                                             int32
         16 engine-size
                              204 non-null
                                             int64
         17 fuel-system
                              204 non-null
                                             int32
         18 bore
                              204 non-null
                                             int32
         19 stroke
                              204 non-null
                                             int32
         20 compression-ratio 204 non-null
                                             float64
         21 horsepower
                              204 non-null
                                             int32
         22 peak-rpm
                              204 non-null
                                             int32
        23 city-mpg
                              204 non-null
                                             int64
         24 highway-mpg
                              204 non-null
                                             int64
         25 price
                              204 non-null
                                             int32
        dtypes: float64(5), int32(16), int64(5)
       memory usage: 28.8 KB
In [ ]:
        import seaborn as sns
        plt.figure(figsize=(20,20))
        sns.heatmap(data clean.corr(),annot=True)
        plt.show()
```

```
-100
                   symboling - 1 0.062 0.1 0.19 0.056 0.64 0.59 0.053 0.22 0.53 0.36 0.23 0.53 0.23 0.077 0.2 0.11 0.085 0.06 0.059 0.18 0.17 0.28 0.031 0.04 0.005
                                     01 022 013 0.098 0072 024 015 029 029 03 011 037 0.044 0.063 033 03 016 0.0014 02 0.013 0.15 0.21 0.23 0.11
              normailsed-losses
                                         0.11 0.05 0.12 0.062 0.0074 0.054 0.063 0.12 0.0043 0.22 0.024 0.081 0.042 0.071 0.16 0.23 0.26 0.14 0.086 0.22 0.048 0.045 0.065
                                                                                                                                                                0.75
                           0.19 -0.22 -0.11 1
                                             0.4 0.21 0.15 0.13 0.04 0.31 0.21 0.23 0.28 0.22 0.089 0.11 0.07 0.04 0.03 0.28 0.98 0.085
                    fuel-type -
                                                  0.064 0.057 0.07 0.058 0.26 0.23 0.3 0.084 0.33 0.11 0.13 0.11 0.29 0.19 0.2 0.3 0.24 0.19 0.2 0.26 0.16
                 rum-of-doors - 0.64 0.098 0.12 0.21 0.064 1 0.067 0.11 0.14 0.41 0.36 0.18 0.51 0.18 0.078 0.15 0.00610.0051 0.05 0.042 0.19 0.13 0.23 0.0034 0.025 0.045
                                                           - 0.50
                           0.053 0.24 0.0074 -0.13 0.07 0.11 -0.14 1
                                                                0.15  0.47  0.49  0.48  0.0078  0.58  0.1  0.22  0.53  0.42  0.5
                           0.22 0.15 0.054 0.04 0.058 0.14 0.29 0.15 1 0.19 0.051 0.052 0.11 0.05 0.11 0.14 0.2 0.11 0.17 0.16 0.02 0.0057 0.19 0.15 0.1 0.012
               engine-locations
                           .0.53 0.29 0.063 -0.31 0.26 -0.41 0.39 0.47 -0.19 1 0.88 0.8 0.58 0.78 -0.16 -0.19 0.57 0.4 0.46 0.15 0.25 0.12 0.37 -0.48 -0.55 -0.33
                                                                                                                                                               - 0.25
                            0.36 0.29 0.12 0.21 0.23 0.36 0.34 0.49 0.051 0.88 1 0.84 0.49 0.88 0.12 0.11 0.68 0.56 0.58 0.12 0.16 0.24 0.28 0.67 0.71 0.4
                     length
                                   .053 011 022 0.28 0084 0.51 0.56 0.0078 0.11 0.58 0.49 027 1 03 0.16 0.29 0.069 0.026 0.12 0.11 0.26 0.13 0.33 0.056 0.11 0.14
                                                                                                                                                               - 0.00
                           .0.23 0.37 0.024 0.22 0.33 0.18 0.13 0.58 0.05 0.78 0.88 0.87 0.3 1 0.057 0.047 0.85 0.61 0.62 0.15 0.15 0.15 0.31 0.27 0.76 0.8 0.41
                            0077 - 0.044 - 0.081 - 0.089 - 0.11 - 0.078 - 0.083 - 0.1 - 0.11 - 0.15 - 0.12 0.00044 - 0.16 - 0.057 - 1 - 0.24 - 0.043 - 0.081 - 0.15 - 0.019 - 0.078 - 0.068 - 0.037 - 0.096 - 0.088 - 0.05
                            02 0.063 0.042 0.11 0.13 0.15 0.052 0.22 0.14 0.19 0.11 0.2 0.29 0.047 0.24 1 0.086 0.013 0.16 0.11 0.065 0.13 0.21 0.13 0.086 0.2
                           0.11 0.33 0.071 -0.07 0.11 0.0061-0.074 0.53 02 0.57 0.68 0.74 0.069 0.85 0.043 0.086 1 0.51 0.52 0.12 0.029 0.24 0.24 0.65 0.68 0.24
                                                                                                                                                                -0.25
                            0.085 0.3 0.16 0.04 0.29 0.0051-0.053 0.42 0.11 0.4 0.56 0.53 0.026 0.61 0.081 0.013 0.51
                                                                                                                 0.081 -0.1 -0.55 0.026 -0.67 -0.64 -0.37
                           -0.06 0.16 0.23 -0.03 0.19 -0.05 -0.011 0.5 0.17 0.46 0.58 0.53 0.12 0.62 0.16 0.16 0.52 0.45
                           0.059 0.0014 0.26 0.28 0.2 0.042 0.062 0.12 0.16 0.15 0.12 0.18 0.11 0.15 0.019 0.11 0.12 0.081 0.029 1 0.23 0.051 0.014 0.087 0.086 0.073
                                                                                                                                                               -0.50
                           -0.18 0.2 0.14 <del>-0.98 0.3 -0.19 0.14 0.13 -0.02 0.25 0.16 0.18 0.26 0.15 -0.078 0.065 0.029 -0.1 -0.014 0.23 1</del>
                           0.17 0.013 0.086 0.085 0.24 0.13 0.12 0.28 0.0057 0.12 0.24 0.25 0.13 0.31 0.068 0.13 0.24 0.55 0.24 0.051 0.1 1 0.23
                                             0.19 0.23 0.11 0.076 0.19 0.37 0.28 0.21 0.33 0.27 0.037 0.21 0.24 0.026 0.18 0.014 0.44 0.23 1
                                                                                                                                                               - -0.75
                           0.031 -0.21 0.048 -0.26 -0.2 0.0034 0.023 -0.45 -0.15 -0.48 -0.67 -0.65 -0.056 -0.76 -0.096 -0.13 -0.65 -0.67 -0.62 -0.087 -0.32 -0.45 -0.12 -1 0.97
                           0.04 0.23 0.045 0.19 0.26 0.025 0.016 0.45 0.1 0.55 0.71 0.68 0.11 0.8 0.088 0.086 0.68 0.64 0.62 0.086 0.26 0.44 0.045 0.97 1
                            .0097-0.11 0.063 0.083 0.16 0.045 0.013 0.32 0.012 0.33 0.4 0.34 0.14 0.41 0.05 0.2 0.24 0.37 0.36 0.073 0.06 0.31 0.089
In [ ]:
                drop_cols = ['symboling','stroke','num-of-doors','body-style','engine-type','compression
                data clean = data clean.drop(drop cols,axis = 1)
In [ ]:
                data_clean.info()
              <class 'pandas.core.frame.DataFrame'>
              RangeIndex: 204 entries, 0 to 203
              Data columns (total 20 columns):
                       Column
                                                       Non-Null Count
                                                                                  Dtype
                0
                      normailsed-losses 204 non-null
                                                                                  int32
                1
                      make
                                                       204 non-null
                                                                                  int32
                2
                      fuel-type
                                                       204 non-null
                                                                                  int32
                3
                       aspiration
                                                       204 non-null
                                                                                   int32
                4
                       drive-wheels
                                                       204 non-null
                                                                                  int32
                5
                       engine-locations
                                                       204 non-null
                                                                                   int32
```

```
wheel-base
                                204 non-null
                                                float64
         6
         7
             length
                                204 non-null
                                                float64
         8
             width
                                204 non-null
                                                float64
         9
             height
                                204 non-null
                                                float64
         10 curb-weight
                                204 non-null
                                                int64
         11 num-of-cylinders
                                204 non-null
                                                int32
         12 engine-size
                                204 non-null
                                                int64
         13 fuel-system
                                204 non-null
                                               int32
         14 bore
                                204 non-null
                                                int32
         15 horsepower
                                204 non-null
                                                int32
         16 peak-rpm
                                204 non-null
                                                int32
         17 city-mpg
                                204 non-null
                                                int64
         18 highway-mpg
                                204 non-null
                                                int64
         19 price
                                204 non-null
                                                int32
        dtypes: float64(4), int32(12), int64(4)
        memory usage: 22.4 KB
In [ ]:
         X = data clean.values[:,:-1]
         Y = data clean.values[:,-1]
         print(X.shape,Y.shape)
        (204, 19) (204,)
In [ ]:
         u = np.mean(X,axis=0)
         std = np.std(X,axis = 0)
         X = (X-u)/std
In [ ]:
         ones = np.ones((X.shape[0],1))
         X = np.hstack((ones,X))
         print(X.shape)
        (204, 20)
In [ ]:
         from sklearn.model selection import train test split
         X train, X test, Y train, Y test = train test split(X,Y,test size=0.1,random state = 7)
In [ ]:
         def r2(Y,Y_):
             num = np.sum((Y-Y)**2)
             denom = np.sum((Y-Y.mean())**2)
             score = 1-(num/denom)
             return score*100
         def train val split(X,Y):
             X_train_new,X_val,Y_train_new,Y_val = train_test_split(X,Y,test_size = 0.05, shuffl
             return X_train_new,X_val,Y_train_new,Y_val
In [ ]:
         def hypothesis(X,theta):
             return np.dot(X,theta)
         def error(X,y,theta):
             e = 0.0
             y_ = hypothesis(X,theta)
             e = np.sum((y_-y)**2)
             return e/X.shape[0]
```

```
def gradient(X,y,theta):
    m = X.shape[0]
    y_ = hypothesis(X,theta)
    grad = np.dot(X.T,(y_-y))
    return grad/m
def gradient_descent(X,y,lr=0.1):
    errors = []
    n = X.shape[1]
    theta = np.zeros((n,))
    val_scores = []
    for i in range(300):
        X_train_new,X_val,Y_train_new,Y_val = train_val_split(X,y)
        e = error(X,y,theta)
        errors.append(e)
        y_ = hypothesis(X_val,theta)
        score = r2(Y val, y)
        val scores.append(score)
        grad = gradient(X_train_new,Y_train_new,theta)
        theta = theta - lr*grad
    return theta, errors, val scores
```

```
In [ ]:
    theta,errors,val_scores = gradient_descent(X_train,Y_train)
    plt.plot(errors)
    plt.show()
```



```
In [ ]: val_score = pd.DataFrame(np.array(val_scores),columns=['Val Score'])
    print(val_score.describe())
Val Score
```

count 300.000000 mean 19.133166 std 39.319306 min -286.155864

```
25% 11.087503

50% 25.716911

75% 40.188743

max 80.213723

In []: y_ = hypothesis(X_test,theta)

print("R2-score: ",r2(Y_test,y_))

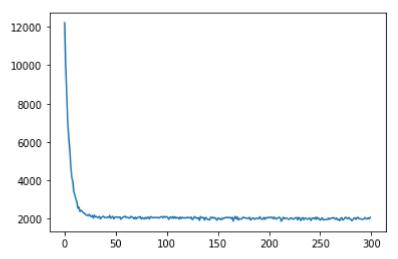
R2-score: 25.30631229492608
```

Lasso Regression

```
In [ ]:
         def hypothesis lasso(X,theta):
             return np.dot(X,theta)
         def error_lasso(X,y,theta,lamda):
             e = 0.0
             y = hypothesis lasso(X,theta)
             e = np.sum((y - y)**2) + lamda*np.sum(np.abs(theta))
             return e/X.shape[0]
         def gradient lasso(X,y,theta,lamda):
             m = X.shape[0]
             y_ = hypothesis_lasso(X,theta)
             grad = np.dot(X.T,(y_-y)) + lamda*(2*np.sinc(theta)-1)
             return grad/m
         def gradient_descent_lasso(X,y,lamda,lr=0.1):
             errors = []
             n = X.shape[1]
             theta = np.zeros((n,))
             val scores = []
             for i in range(300):
                 X_train_new,X_val,Y_train_new,Y_val = train_val_split(X,y)
                 e = error_lasso(X_train_new,Y_train_new,theta,lamda)
                 errors.append(e)
                 y_ = hypothesis_lasso(X_val,theta)
                 score = r2(Y_val,y_)
                 val_scores.append(score)
                 grad = gradient_lasso(X_train_new,Y_train_new,theta,lamda)
                 theta = theta - lr*grad
             return theta,errors,val_scores
```

```
In [ ]: theta_lasso,error_lasso,val_score_lasso = gradient_descent_lasso(X_train,Y_train,10)
In [ ]: plt.plot(error_lasso)
    plt.show()
```

10/6/21, 11:52 PM linearRegression



min -428.959032 25% 11.039810 50% 27.642138 75% 39.491201 max 69.706456

```
In [ ]: 
y_ = hypothesis_lasso(X_test,theta_lasso)
print("R2-score: ",r2(Y_test,y_))
```

R2-score: 25.196130078133372

Ridge Regression

```
In [ ]:
         def hypothesis_lasso(X,theta):
             return np.dot(X,theta)
         def error_lasso(X,y,theta,lamda):
             e = 0.0
             y_ = hypothesis_lasso(X,theta)
             e = np.sum((y_-y)**2) + lamda*np.sum(np.square(theta))
             return e/X.shape[0]
         def gradient_lasso(X,y,theta,lamda):
             m = X.shape[0]
             y_ = hypothesis_lasso(X,theta)
             grad = np.dot(X.T,(y_-y)) + lamda*theta
             return grad/m
         def gradient_descent_lasso(X,y,lamda,lr=0.1):
             errors = []
             n = X.shape[1]
             theta = np.zeros((n,))
```

```
val_scores = []
             for i in range(300):
                 X_train_new,X_val,Y_train_new,Y_val = train_val_split(X,y)
                  e = error_lasso(X_train_new,Y_train_new,theta,lamda)
                  errors.append(e)
                 y_ = hypothesis_lasso(X_val,theta)
                  score = r2(Y_val,y_)
                 val scores.append(score)
                  grad = gradient_lasso(X_train_new,Y_train_new,theta,lamda)
                  theta = theta - lr*grad
             return theta,errors,val_scores
In [ ]:
         theta ridge,error ridge,val score ridge = gradient descent lasso(X train,Y train,5)
In [ ]:
         plt.plot(error ridge)
         plt.show()
         12000
         10000
          8000
          6000
          4000
          2000
                       50
                              100
                                      150
                                             200
                                                    250
                                                            300
In [ ]:
         val_score_ridge = pd.DataFrame(np.array(val_score_ridge),columns=['Val Score Ridge'])
         print(val_score_ridge.describe())
                Val Score Ridge
                     300.000000
        count
                       9.630181
        mean
        std
                      79.710641
                   -1038.155673
        min
        25%
                       1.371537
        50%
                      21.629123
        75%
                      39.950043
                      72.980039
        max
In [ ]:
         y_ = hypothesis_lasso(X_test,theta_ridge)
         print("R2-score: ",r2(Y_test,y_))
```

R2-score: 24.232403338944618

In []: