Supervised Machine Learning: Regression

Peer To Peer Assignment

Main objective: Car Price Prediction

We are required to model the price of cars with the available independent variables. It will be used by the management to understand how exactly the prices vary with the independent variables. They can accordingly manipulate the design of the cars, the business strategy etc. to meet certain price levels.

Brief description of the data set and a summary of its attributes

I have data set consisting of 205 data points and 26 columns representing features.

There are some features that are not important for target variable , I will simply drop them and there are also some categorical feature, that I will convert to numerical form by Encoding method.

car_ID ID of every car

CarName Name of Car

fueltype Type of Fuel

doornumber Total number of door

carbody Body of Car weather Sedan or Hatchback etc

enginelocation Location of engine in car

wheelbase Distance between rear and front wheel

carlength Length of Car

carwidth Width of Car

carheight Height of Car

curbweight Weight of Car without any passenger or item

enginetype Type of engine

cylindernumber Total cylinder in Car

enginesize Size of Engine

boreratio Combustion Performance of Lean Burn Heavy-Duty Gaseous Engine

stroke A phase of the engine's cycle

horsepower Power of Car

peakrpm Revolution per minute

citympg City mileage per gallon

highwaympg Highway mileage per gallon

price Price of car

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df = pd.read_csv('E:\Machine Learning Course\Course2\week3\CarPrice.csv')
```

```
df.head()
```

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...	enginesize	fuelsystem	boreratio	s
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	
3	4	2	audi 100 ls	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	
4	5	2	audi 100ls	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	

5 rows × 26 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   car_ID            205 non-null    int64
 1   symboling         205 non-null    int64
 2   CarName           205 non-null    object
 3   fueltype          205 non-null    object
 4   aspiration        205 non-null    object
 5   doornumber        205 non-null    object
 6   carbody           205 non-null    object
 7   drivewheel        205 non-null    object
 8   enginelocation    205 non-null    object
 9   wheelbase         205 non-null    float64
 10  carlength         205 non-null    float64
 11  carwidth          205 non-null    float64
 12  carheight         205 non-null    float64
 13  curbweight        205 non-null    int64
 14  enginetype        205 non-null    object
 15  cylindernumber    205 non-null    object
 16  enginesize        205 non-null    int64
 17  fuelsystem        205 non-null    object
 18  boreratio         205 non-null    float64
 19  stroke            205 non-null    float64
 20  compressionratio  205 non-null    float64
 21  horsepower        205 non-null    int64
 22  peakrpm           205 non-null    int64
```

```
df.isnull().sum()
```

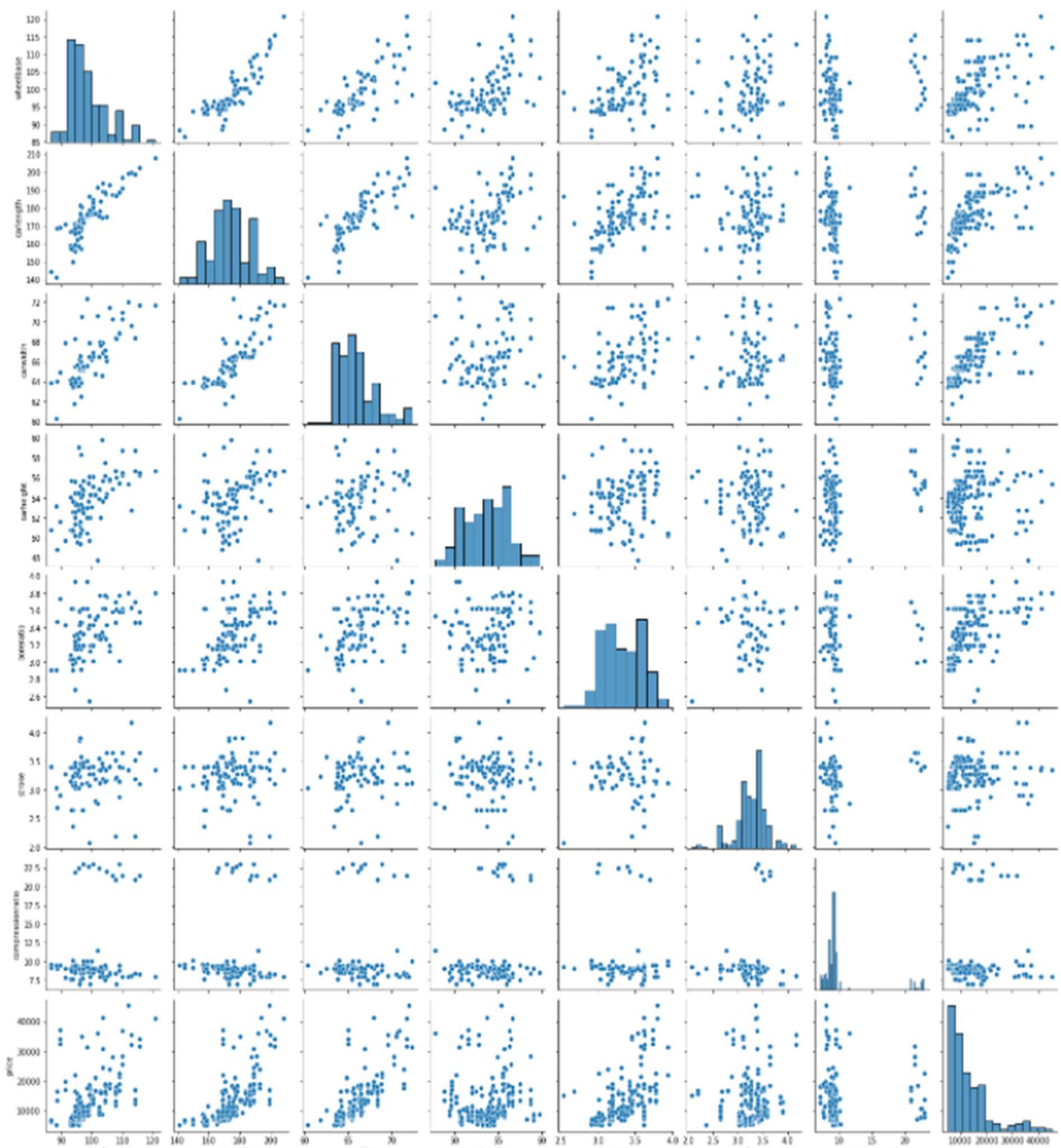
```
car_ID              0
symboling           0
CarName             0
fueltype            0
aspiration          0
doornumber          0
carbody             0
drivewheel          0
enginelocation      0
wheelbase           0
carlength           0
carwidth            0
carheight           0
curbweight          0
enginetype          0
cylindernumber      0
enginesize          0
fuelsystem          0
boreratio           0
stroke              0
compressionratio    0
horsepower          0
peakrpm             0
citympg             0
highwaympg          0
price               0
dtype: int64
```

```python
df.drop(['fuelsystem','cylindernumber','enginetype','aspiration','symboling','car_ID'],inplace=True,axis=1)
```

```python
small_df = df.loc[:,['wheelbase','carlength','carwidth','carheight','boreratio','stroke','compressionratio','price']]
```

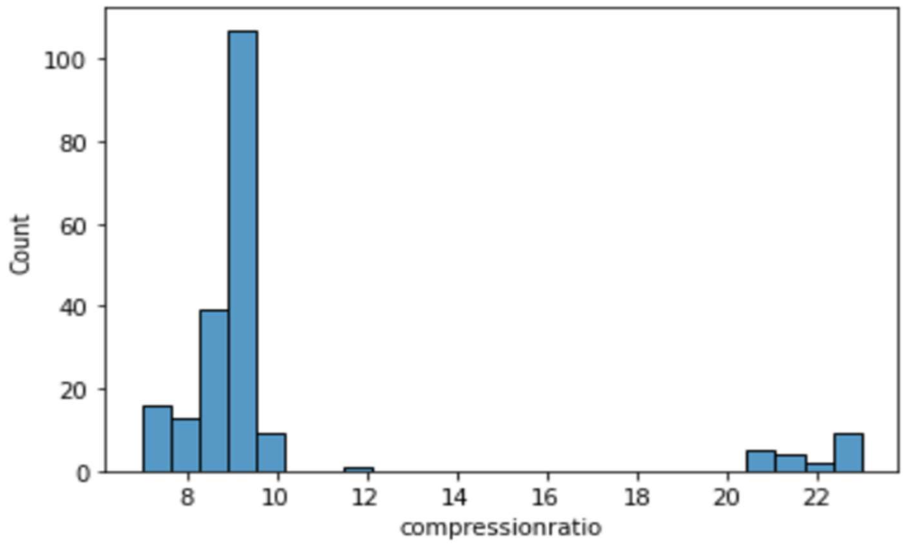
```python
sns.pairplot(small_df)
```

```
<seaborn.axisgrid.PairGrid at 0x23446494b80>
```



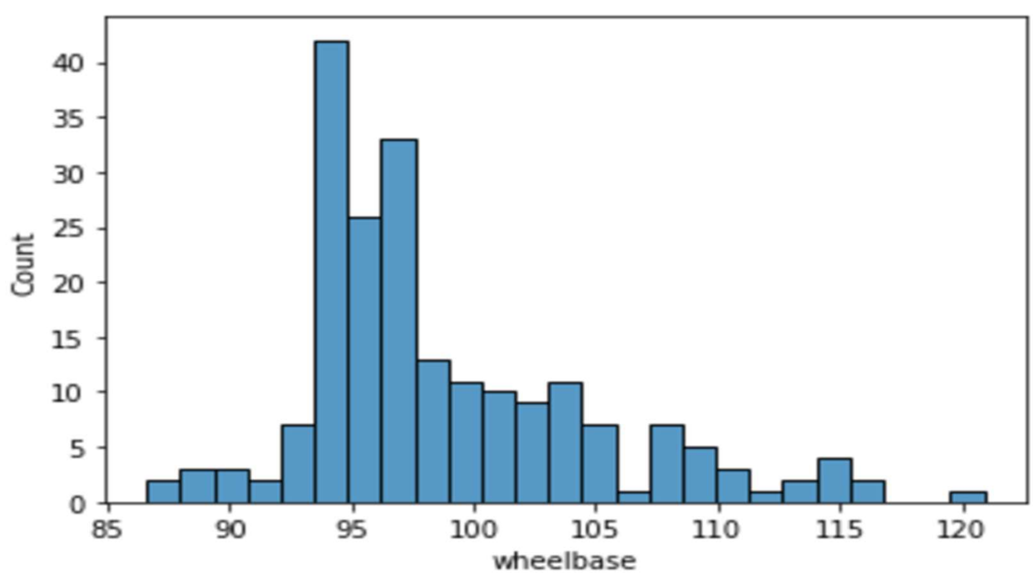
```
sns.histplot(df['compressionratio'],bins=25)
```

<AxesSubplot:xlabel='compressionratio', ylabel='Count'>




```
sns.histplot(df['wheelbase'],bins=25)
```

<AxesSubplot:xlabel='wheelbase', ylabel='Count'>



```python
mask = df.dtypes == np.float
float_cols = df.columns[mask]

skew_limit = 0.75 # define a limit above which we will log transform
skew_vals = df[float_cols].skew()
skew_vals
```

```
<ipython-input-141-30ffafa34c1c>:1: DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To silence th
is warning, use `float` by itself. Doing this will not modify any behavior and is safe. If you specifically wanted the numpy sc
alar type, use `np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  mask = df.dtypes == np.float
```

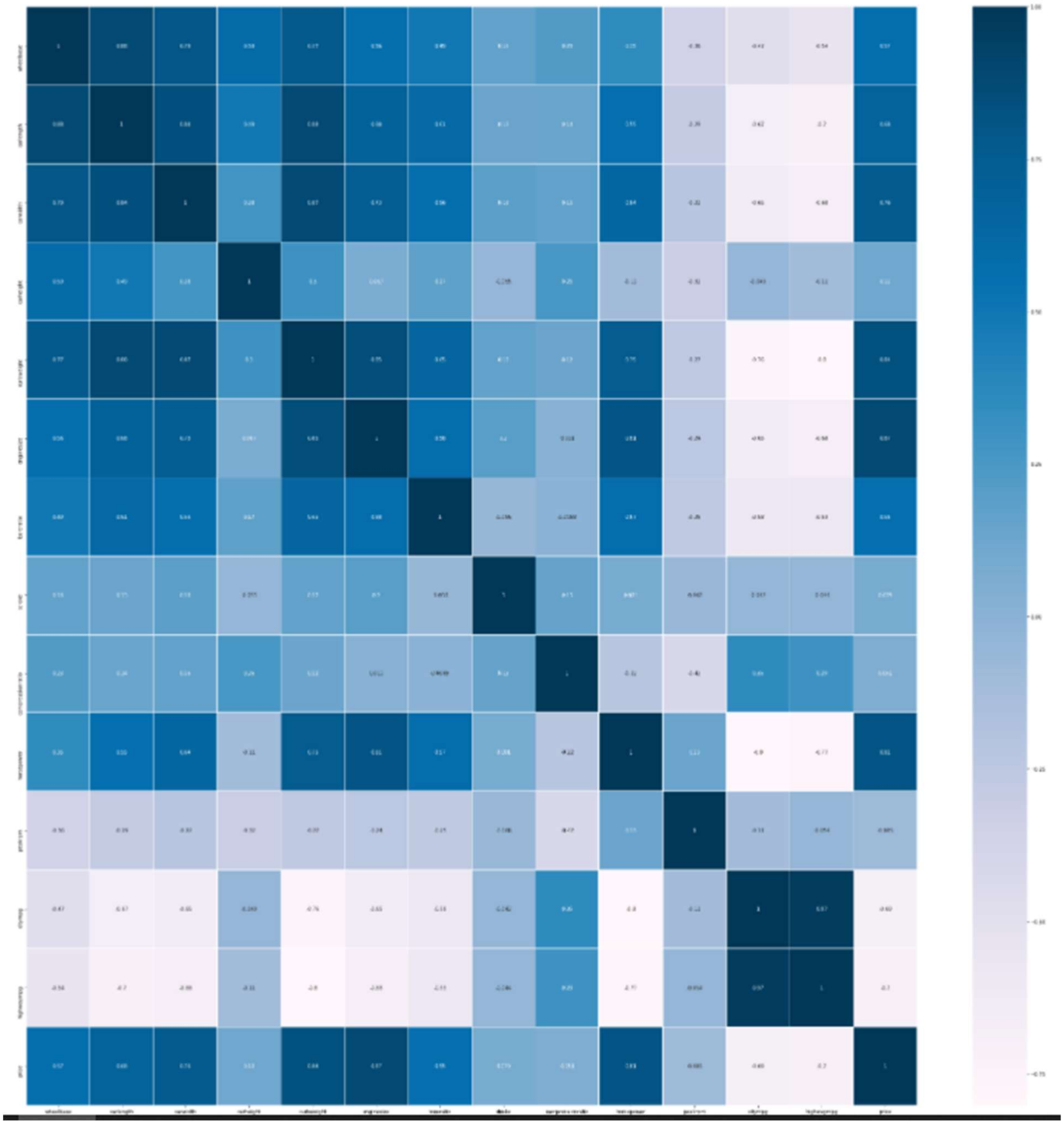
```
wheelbase           1.050214
carlength           0.155954
carwidth            0.904003
carheight           0.063123
boreratio           0.020156
stroke             -0.689705
compressionratio    2.610862
price               1.777678
dtype: float64
```

```python
skew_cols = (skew_vals.sort_values(ascending=False).to_frame().rename(columns={0:'Skew'}).query('abs(Skew) > {}'.format(skew_limi
skew_cols
```

	Skew
compressionratio	2.610862
price	1.777678
wheelbase	1.050214
carwidth	0.904003

```python
for col in skew_cols.index.values:
    if col == "price":
        continue
    df[col] = df[col].apply(np.log1p)
```

```
plt.figure(figsize=(40,40))
sns.heatmap(df.corr(method ='pearson'),cmap='PuBu',annot=True,linewidths=0.5)
plt.show()
```



```
feature = df.dtypes == np.object
```

```
<ipython-input-132-3fa00075be2d>:1: DeprecationWarning: `np.object` is a deprecated a
this warning, use `object` by itself. Doing this will not modify any behavior and is
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/re
  feature = df.dtypes == np.object
```

```
feature = df.dtypes[df.dtypes == np.object]
feature = feature.index.tolist()
```

```
<ipython-input-133-82993708c8bb>:1: DeprecationWarning: `np.object` is a deprecated a
this warning, use `object` by itself. Doing this will not modify any behavior and is
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/re
  feature = df.dtypes[df.dtypes == np.object]
```

```
df_dum = pd.get_dummies(df,columns=feature,drop_first=True)
```

```
df = df.drop(df[feature],axis=1)
```

```
pd.concat([df,df_dum],axis=1)
```

Summary of data exploration and Actions taken for Data Cleaning and Feature Engineering

1) **FINDINGS**
   After reading data by pandas read_csv function, I applied
   a. **isnull()** to see is their any missing in the dataset. But I found their no missing value in my dataset.
   b. **Info()** function to see data type of different feature of my dataset and I found there are many feature that are important for my target variable and their data type is object.
   c. **Skew() and hist()** function to check shewness of data and their was some columns that are right or left skewed. e.g compressionratio and wheel etc.
   d. **Pairplot()** to see correlation and also to find is their any need to use **Polynomial feature** for higher degree. i.e 2,3,4 etc.
   e. **Heatmap()** to see correlation in more depth by printing their corresponding value of relation with each other.
2) **ACTION:**

a. For column that are important for by target variable and their data type was Object, I applied **get_dummies()** function of pandas to convert them into numeric type.
b. For removing skewness of different columns, I applied **log1p** transformation function.
c. Although I found there are some columns are correlated with each other but when I applied **Variation Inflation Factor** technique to remove correlation among them, I found that it have negative impact on my r2_score because in this technique some columns are drop for eliminating correlation. So then I trained my model without removing correlation.
d. Also their was some columns that are not important for our target, So I simply drop them.
e. Similarly, I used **Standard Scaler** and **Polynomial feature** technique on dataset before giving it to model for training.

Summary of Three Different Linear Regression Model

1) Simple Linear Regression Model

```
X = df.drop('price',axis=1)
y = df['price']
```

```
from sklearn.preprocessing import StandardScaler, PolynomialFeatures
from sklearn.model_selection import GridSearchCV, train_test_split, KFold
from sklearn.linear_model import LinearRegression,Lasso,Ridge
from sklearn.pipeline import Pipeline
from sklearn.metrics import r2_score
```

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.25,random_state=72018)
```

```
s = StandardScaler()
X_train_s = s.fit_transform(X_train)
X_test_s = s.transform(X_test)
```

```
lr = LinearRegression()
lr = lr.fit(X_train_s,y_train)
pred = lr.predict(X_test_s)
r2_score(pred,y_test)
```

```
0.6009998432027648
```

2) Ridge Regression Model

```
kf = KFold(shuffle=True,random_state=72018,n_splits=3)
```

```
estimator = Pipeline([
    ("Polynomail_feature",PolynomialFeatures()),
    ("scaler",StandardScaler()),
    ("ridge_regression",Ridge())
])

param = {
    'Polynomail_feature__degree':[1,2,3],
    'ridge_regression__alpha':np.geomspace(4,20,30)
}

grid = GridSearchCV(estimator,param,cv=kf)
```

```
grid.fit(X,y)
```

```
GridSearchCV(cv=KFold(n_splits=3, random_state=72018, shuffle=True),
             estimator=Pipeline(steps=[('Polynomail_feature',
                                        PolynomialFeatures()),
                                       ('scaler', StandardScaler()),
                                       ('ridge_regression', Ridge())]),
             param_grid={'Polynomail_feature__degree': [1, 2, 3],
                         'ridge_regression__alpha': array([ 4.        ,  4.22826702,  4.469560
        5.27924796,  5.58051751,  5.89897953,  6.23561514,  6.59146146,
        6.96761476,  7.36523392,  7.78554391,  8.22983963,  8.69948987,
        9.19594151,  9.72072404, 10.27545421, 10.86184103, 11.48169104,
       12.13691388, 12.82952815, 13.56166768, 14.33558803, 15.15367351,
       16.01844446, 16.93256509, 17.89885162, 18.92028098, 20.        ])})
```

```
pred_r = grid.predict(X)
```

```
r2_score(pred_r,y)
```

```
0.8940782687472872
```

```
grid.best_score_, grid.best_params_
```

```
(0.8229369717953522,
 {'Polynomail_feature__degree': 3,
  'ridge_regression__alpha': 12.82952815374728})
```

3) Lasso Regression Model

```
estimator = Pipeline([
    ("Polynomail_feature",PolynomialFeatures()),
    ("scaler",StandardScaler()),
    ("lasso_regression",Lasso(max_iter=100000))
])

param = {
    'Polynomail_feature__degree':[1,2,3],
    'lasso_regression__alpha':np.geomspace(0.001,10,5)
}

grid_l = GridSearchCV(estimator,param,cv=kf)
```

```
grid_l.fit(X,y)
```

```
GridSearchCV(cv=KFold(n_splits=3, random_state=72018, shuffle=True),
             estimator=Pipeline(steps=[('Polynomail_feature',
                                        PolynomialFeatures()),
                                       ('scaler', StandardScaler()),
                                       ('lasso_regression',
                                        Lasso(max_iter=100000))]),
             param_grid={'Polynomail_feature__degree': [1, 2, 3],
                         'lasso_regression__alpha': array([1.e-03, 1.e-02, 1.e-
```

```
pred_l = grid_l.predict(X)
```

```
r2_score(pred_l,y)
```

```
0.895975904352647
```

```
grid_l.best_score_, grid_l.best_params_
```

```
(0.8177757224961648,
 {'Polynomail_feature__degree': 2, 'lasso_regression__alpha': 10.0})
```

1) As in first case I applied simple linear regression on model and it’s r2_score was 0.60 which is not considered as good result.

2) In second case I applied Polynomial feature technique on dataset before giving it to Ridge regression. I also used GridSearchCV method and pipeline technique to make process fast and easy to find out best best parameters for our model to have good prediction.

So by doing this, I found that parameters i.e polynomial feature and alpha values 3 and 14 are best for our model to have r2_score of 0.89 .

3) Similarly for case three I used Lasso Regression, GridSearchCV and pipeline technique and founded that value of 2 for degree and 10 for alpha is best for our model to have good r2_score of 0.89 .

A paragraph explaining which of your regressions you recommend as a final model that best fits your needs in terms of accuracy and explainability.

For choosing model that best fir our data, Ridge and Lasso both are better than simple linear regression and both have same r2_score, So we can choose any one from both of them but if we want interpretability along with our main goal of prediction then Lasso will be choice.

Summary Key Findings and Insights, which walks your reader through the main drivers of your model and insights from your data derived from your linear regression model

Some of key point for this data and regression model is that as we know that correlation in data is not good for our model and it have impact on model accuracy but in our case when we used technique of VIF to eliminate correlation from data, I found that it does not have good impact on accuracy of model because in this technique we drop one of column that are correlated with each other, So by doing that we will end up with having less number of feature for our model to train on which is again problem. i.e Problem of underfitting.

Suggestions for next steps in analyzing this data, which may include suggesting revisiting this model adding specific data features to achieve a better explanation or a better prediction

So my next suggestion for analyzing this data will be to have more data because as we can see our data consist of only 205 row or observation that is not good enough to train model.