Predict the insurance charges of a person based on factors like age, bmi, smoking status, children,etc.

The data set is collected from Kaggle. The size of the sample is N= 1338 and 7 variables.

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

We describe the attributes of the data.

```
data.describe()
```

	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	13270.422265
std	14.049960	6.098187	1.205493	12110.011237
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.296250	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	9382.033000
75%	51.000000	34.693750	2.000000	16639.912515
max	64.000000	53.130000	5.000000	63770.428010

```
data.dtypes
```

```
age           int64
sex          object
bmi         float64
children      int64
smoker       object
region       object
charges     float64
dtype: object
```

Initial Plans for data exploration:

Firstly, we will clean our data by removing the null variables. Then, perform Univariate and Bivariate analysis of the attributes. Feature engineering is also important in order to get accurate results after fully exploring the data. We will encode the categorical variables. We will also have to perform hypothesis testing so as to analyse the correlation between explanatory variables and target variable.

All these steps are part of Exploratory Data Analysis. EDA is important for every data set before developing any ML model so as to make the data clean and ready for modelling to get accurate results.

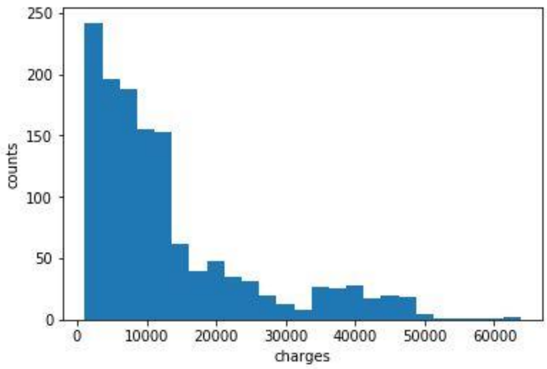
Univariate Analysis

All the variables of the data are visually analysed with the help of graphs and plots.

Univariate Analysis

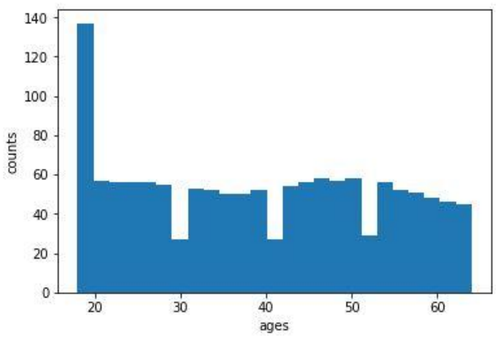
```
#Visualization of target variable (charges)
plt.hist(x=data.charges,bins=25)
plt.xlabel('charges')
plt.ylabel('counts')
```

```
Text(0, 0.5, 'counts')
```



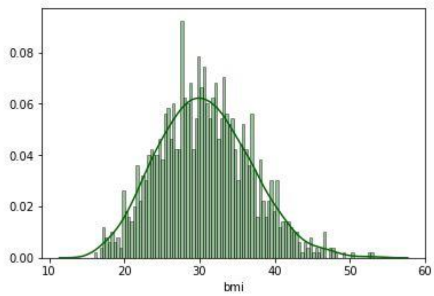

```
#visualizing explanatory variables
plt.hist(x=data.age,bins=25)
plt.xlabel('ages')
plt.ylabel('counts')
```

```
Text(0, 0.5, 'counts')
```



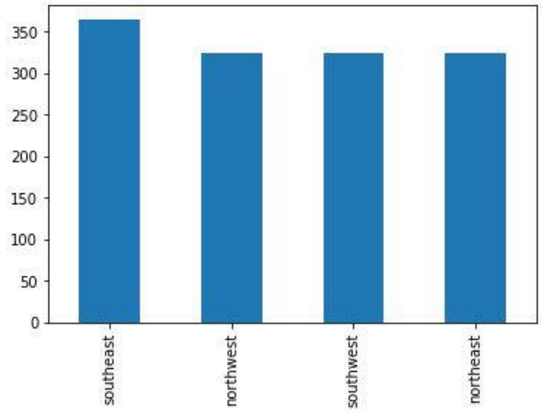

```
sns.distplot(data['bmi'],hist = True, color = 'darkgreen', bins = 100, hist_kws={'edgecolor':'black'})
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff4f5d7be80>
```



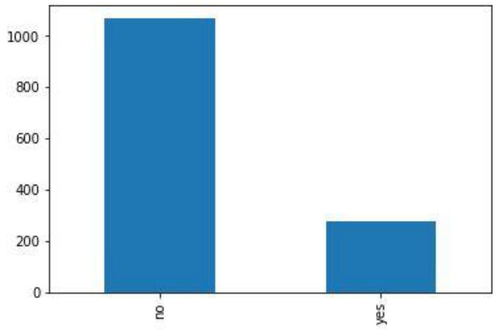
```
data['region'].value_counts().plot.bar()
```

<matplotlib.axes._subplots.AxesSubplot at 0x7ff4f5d38828>



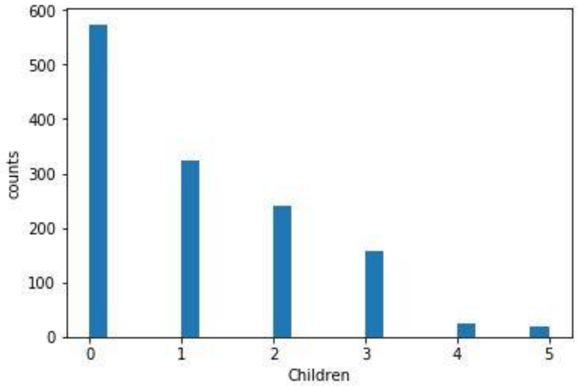

```
data['smoker'].value_counts().plot.bar()
```

<matplotlib.axes._subplots.AxesSubplot at 0x7ff4f5bb6d30>




```
plt.hist(x=data.children,bins=25)
plt.xlabel('Children')
plt.ylabel('counts')
```

Text(0, 0.5, 'counts')



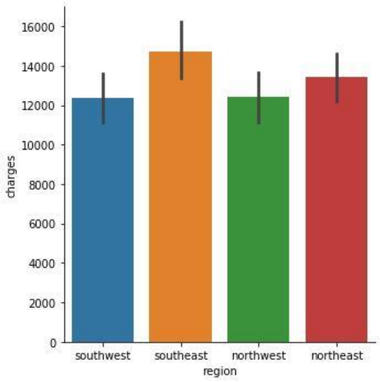
Bivariate Analysis

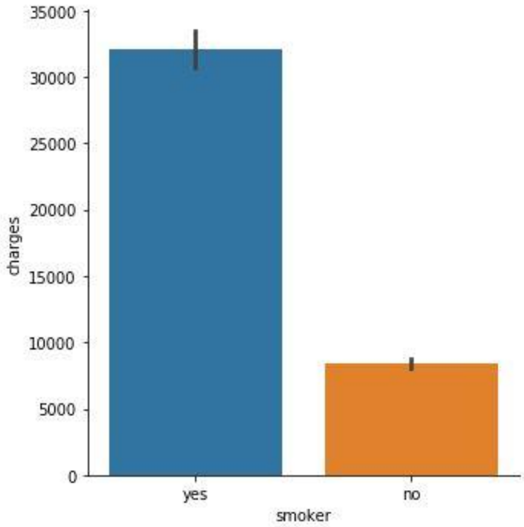
This is for analysing the relationship between explanatory variables and target variable.

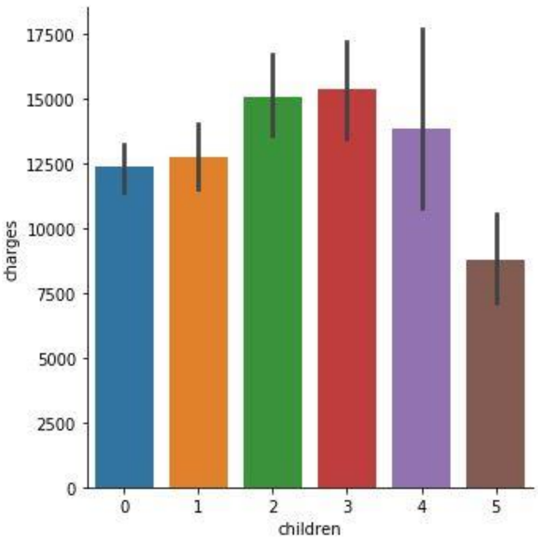
```
sns.factorplot(x='region',y='charges',data=data,kind='bar')
plt.xlabel('region')
plt.ylabel('charges')
```

```
/opt/conda/envs/Python36/lib/python3.6/site-packages/seaborn/categorical.py:3666: Us
en renamed to `catplot`. The original name will be removed in a future release. Plea
`kind` in `factorplot` (`'point'`) has changed `'strip'` in `catplot`.
  warnings.warn(msg)
```

```
Text(-8.825000000000003, 0.5, 'charges')
```

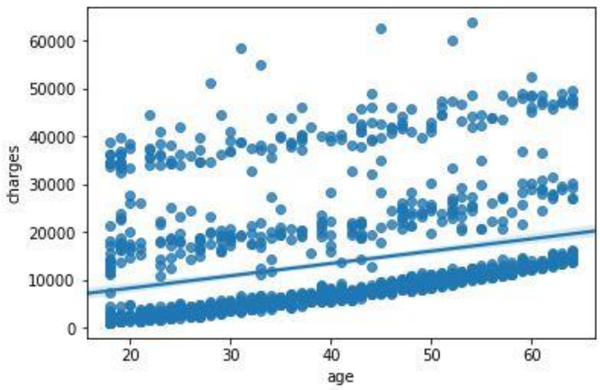






```
scat=sns.regplot(x='age',y='charges',data=data)
plt.xlabel('age')
plt.ylabel('charges')
```

```
Text(0, 0.5, 'charges')
```



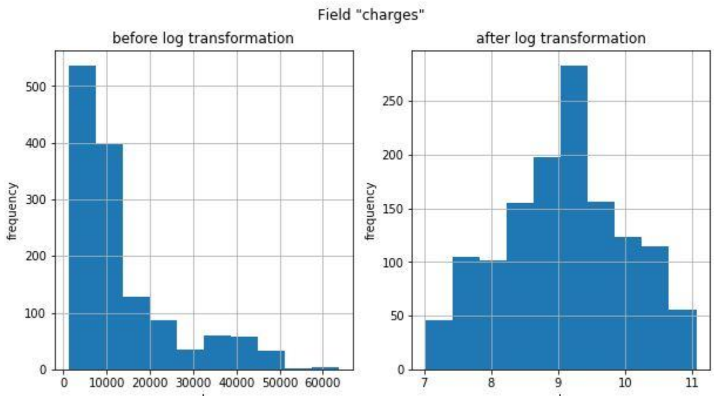
Feature Engineering

It is a critical phase in data exploration and EDA. The variables are modified as per the requirements and cleaned in order to get better and accurate results.

Firstly, we normalize the variables which do not show normal distributions. Normalized variables provide good results. We do this by performing log transformation on the variables.

```
#Log transforming the skew variables
num_cols=data.select_dtypes('number').columns
skew_limit=0.75
skew_values=data.skew()
```

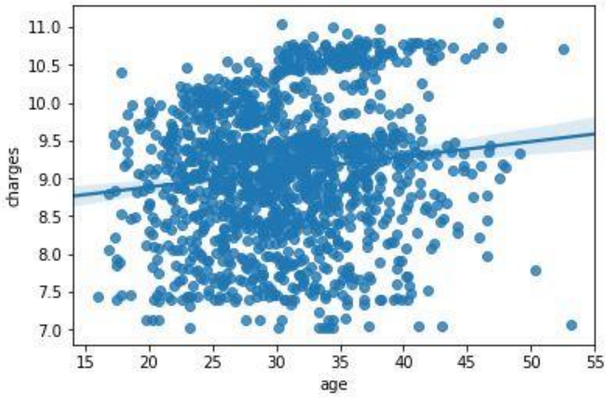
```
field='charges'
fig,(ax_before,ax_after)=plt.subplots(1,2, figsize=(10,5))
data[field].hist(ax=ax_before)
data[field].apply(np.log1p).hist(ax=ax_after)
ax_before.set(title='before log transformation',ylabel='frequency',xlabel='value')
ax_after.set(title='after log transformation',ylabel='frequency',xlabel='value')
fig.suptitle('Field "{}"'.format(field));
```



Next, we can again display a regression plot to see the difference after log transformation.

```
scat=sns.regplot(x='bmi',y='charges',data=data)
plt.xlabel('age')
plt.ylabel('charges')
```

```
Text(0, 0.5, 'charges')
```

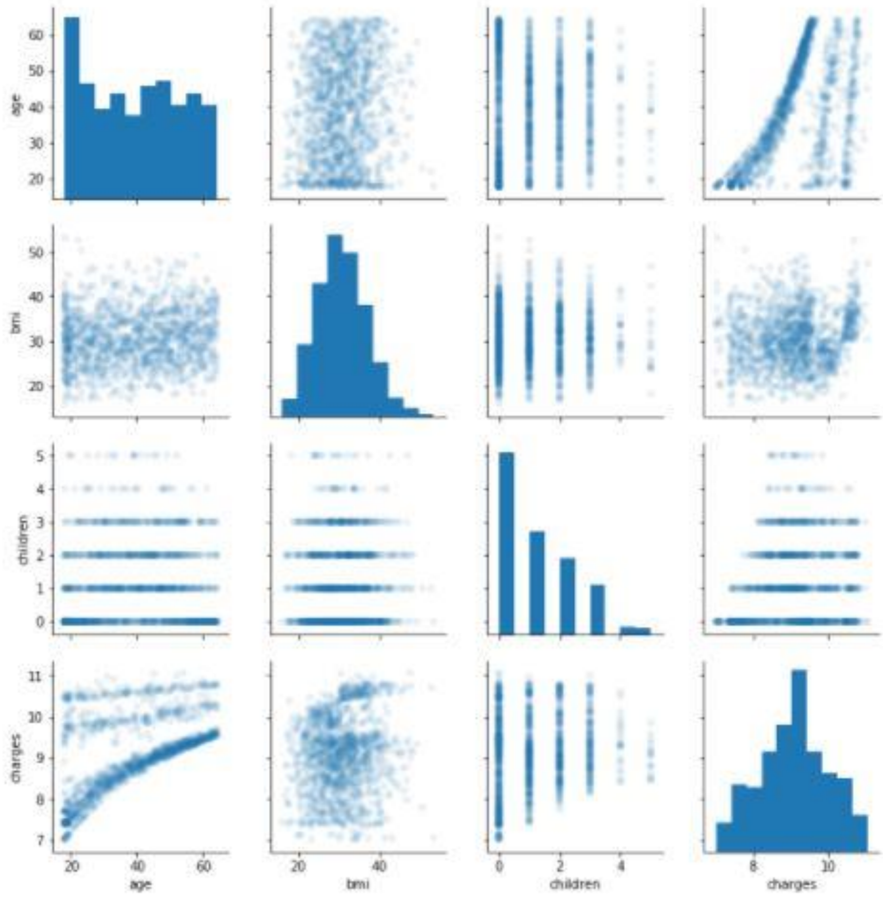


We can see that it shows a better relationship.

We can also display pair plots for all the variables of the data.

```
#Pair plots of features
sns.pairplot(data,plot_kws=dict(alpha=.1,edgecolor='none'))
```

```
<seaborn.axisgrid.PairGrid at 0x7ff4f531da90>
```



Encoding categorical variables:

We need to encode the categorical variables into numeric so that they can be fit into the model.

```
#convert categorical variables with 2 levels to binary variables
from sklearn import preprocessing
le=preprocessing.LabelEncoder()
data['sex']=le.fit_transform(data['sex'])
data['smoker']=le.fit_transform(data['smoker'])
data['region']=le.fit_transform(data['region'])
```

```
data.head()
```

	age	sex	bmi	children	smoker	region	charges
0	19	0	27.900	0	1	3	9.734236
1	18	1	33.770	1	0	2	7.453882
2	28	1	33.000	3	0	2	8.400763
3	33	1	22.705	0	0	1	9.998137
4	32	1	28.880	0	0	1	8.260455

Hypothesis Testing

This is again an important step. It helps us get a clear understanding about our data and the relationship between its variables. It tells us whether our hypothesis is correct or not or whether we should consider the null hypothesis.

For the given data and problem statement, our hypothesis is that there is a correlation between the explanatory variables(age,sex,bmi,children,region,smoker) and the target variable(charges). The null hypothesis for this case would be that there is no relation between the variables.

So, we will check each of the variables separately and see which variables are correlated and which are not.

- To analyse the association between binary explanatory variables and the continuous response variable we use ANOVA.

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                charges   R-squared:                       0.000
Model:                            OLS   Adj. R-squared:                 -0.001
Method:                 Least Squares   F-statistic:                   0.04256
Date:                Tue, 08 Sep 2020   Prob (F-statistic):              0.837
Time:                        17:38:53   Log-Likelihood:                -1785.6
No. Observations:                1338   AIC:                             3575.
Df Residuals:                    1336   BIC:                             3585.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      9.0936      0.036    254.398      0.000       9.023       9.164
sex            0.0104      0.050      0.206      0.837      -0.088       0.109
==============================================================================
Omnibus:                       52.338   Durbin-Watson:                   1.991
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               24.513
Skew:                          -0.092   Prob(JB):                     4.75e-06
Kurtosis:                       2.363   Cond. No.                         2.63
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

p-value for association between 'sex' and 'charges' is 0.837 > 0.05, so it is insignificant. We cannot reject null hypothesis for this case.

The p-value for this is large and is greater than 0.05, so we have to consider null hypothesis and reject the alternate hypothesis. This means that there is no correlation between 'sex' of a person and their insurance charges.

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                charges   R-squared:                       0.443
Model:                            OLS   Adj. R-squared:                  0.443
Method:                 Least Squares   F-statistic:                     1062.
Date:                Tue, 08 Sep 2020   Prob (F-statistic):          5.98e-172
Time:                        17:38:54   Log-Likelihood:                -1394.1
No. Observations:                1338   AIC:                             2792.
Df Residuals:                    1336   BIC:                             2803.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      8.7884      0.021    417.614      0.000       8.747       8.830
smoker         1.5157      0.047     32.593      0.000       1.424       1.607
==============================================================================
Omnibus:                       26.359   Durbin-Watson:                   1.998
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               26.792
Skew:                          -0.327   Prob(JB):                     1.52e-06
Kurtosis:                       2.772   Cond. No.                         2.60
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

p-value is very very small < 0.05, so we can reject the null hypothesis and move forward with the assumption that there is a relation between the two variables

The p-value here is very small than 0.05, so it is significant and means that we can reject the null hypothesis.

- Next, we test for the association between categorical explanatory variables and target variable using Post-hoc test.

```
#post-hoc test for categorical variables with more than 2 levels
import statsmodels.stats.multicomp as multi
```

```
mc1=multi.MultiComparison(data['charges'], data['region'])
res1=mc1.tukeyhsd()
print(res1.summary())
```

```
Multiple Comparison of Means - Tukey HSD,FWER=0.05
=============================================
group1 group2 meandiff  lower  upper  reject
---------------------------------------------
  0      1     -0.099  -0.2846 0.0866 False
  0      2    -0.0463  -0.2269 0.1342 False
  0      3    -0.1376  -0.3232 0.0479 False
  1      2     0.0527  -0.1277 0.2331 False
  1      3    -0.0386  -0.2241 0.1468 False
  2      3    -0.0913  -0.2717 0.0891 False
---------------------------------------------
```

The above results show that we cannot reject the null hypothesis for any of the groups.

This shows that there is no correlation between the considered variables.

- Lastly, for analysing the association between continuous explanatory variables and the target variable we use Pearson correlation coefficient.

```python
#Pearson correlation between continuous explanatory variables and target variable
import scipy.stats
print('association between continuous explanatory variables and target variable')
print('1. Age and Charges')
print(scipy.stats.pearsonr(data['age'],data['charges']))
print('1. BMI and Charges')
print(scipy.stats.pearsonr(data['bmi'],data['charges']))
print('1. Children and Charges')
print(scipy.stats.pearsonr(data['children'],data['charges']))
```

```
association between continuous explanatory variables and target variable
1. Age and Charges
(0.527807363459784, 7.675383914743264e-97)
1. BMI and Charges
(0.13267798192830776, 1.1148799774752429e-06)
1. Children and Charges
(0.16131660389394725, 2.9539064053854775e-09)
```

We can see above that there is a correlation between the given continuous variables and the target variable, so we can reject the null hypothesis.

After analysing all the variables we can clearly say that 'sex' and 'region' have no clear correlation with the target variable 'charges'. So we can remove these variables from our data set.

After all the steps of Feature Engineering and Hypothesis testing, our final data set look like the following:

```python
final_data= data[['age','smoker','bmi','children','charges']]
```

```python
final_data.head()
```

	age	smoker	bmi	children	charges
0	19	1	27.900	0	9.734236
1	18	0	33.770	1	7.453882
2	28	0	33.000	3	8.400763
3	33	0	22.705	0	9.998137
4	32	0	28.880	0	8.260455

The next steps after this is to develop a model for solving the problem question and predict the results.

Regression model is an appropriate model for the given data. We will build different regression models and analyse the results by comparing various models. We will compare the accuracy scores and error metrics and identify the most appropriate model.

Regression Analysis

- Linear Regression

```
#Linear Regression
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import (StandardScaler,PolynomialFeatures)
```

```
lr=LinearRegression()
```

```
X=final_data[['age','smoker','bmi','children']]
y=final_data['charges']
```

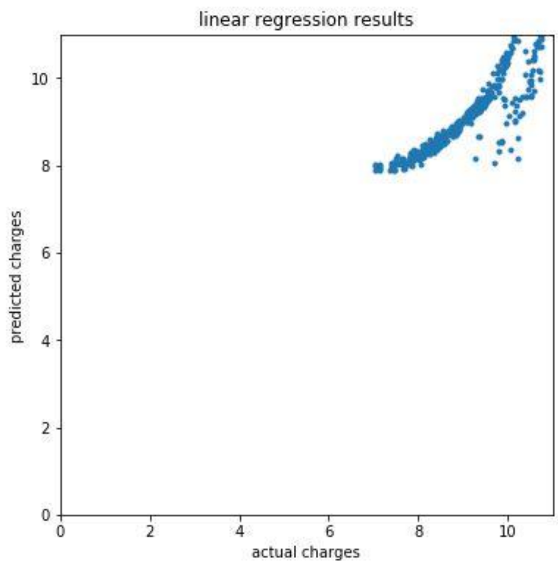
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=1234)
```

```
lr.fit(X_train,y_train)
y_pred=lr.predict(X_test)
y_pred[0:5]
```

```
array([8.82167689, 8.25530661, 8.30804311, 8.66784491, 9.37933152])
```

```
print(y_pred[0:5],y_test[0:5])
```

```
[8.82167689 8.25530661 8.30804311 8.66784491 9.37933152] 395     8.926346
809     8.104943
324     7.961053
237     8.403846
1132    9.938373
Name: charges, dtype: float64
```



Cross-validation:

```
from sklearn.model_selection import KFold,cross_val_predict
from sklearn.pipeline import Pipeline
```

```
#for linear regression
kf=KFold(shuffle=True,random_state=1234,n_splits=3)
```

```
from sklearn.metrics import mean_squared_error
```

```
scores=[]
lr3=LinearRegression()
for train_index,test_index in kf.split(X):
    trainx,testx,trainy,testy=(X.iloc[train_index,:],
                               X.iloc[test_index,:],
                               y[train_index],
                               y[test_index])
    lr3.fit(trainx,trainy)
    y_pr=lr3.predict(testx)
    score=r2_score(testy.values,y_pr)
    scores.append(score)
scores
```

```
[0.7854560153775929, 0.7687688877577215, 0.7198963632889281]
```

- Polynomial Regression

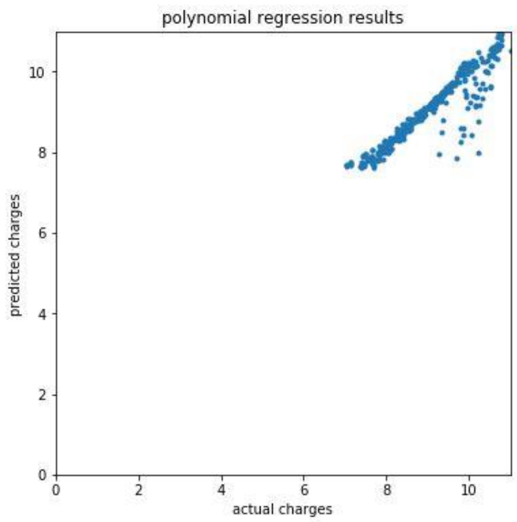
```
#Polynomial Regression
from sklearn.preprocessing import PolynomialFeatures
train_x = np.asanyarray(X_train)
train_y = np.asanyarray(y_train)

test_x = np.asanyarray(X_test)
test_y = np.asanyarray(y_test)

lr2=LinearRegression()
poly = PolynomialFeatures(degree=2)
train_x_poly = poly.fit_transform(train_x)
train_x_poly
```

```
lr2.fit(train_x_poly, train_y)
# The coefficients
print ('Coefficients: ', lr.coef_)
print ('Intercept: ',lr.intercept_)
```

```
Coefficients:  [0.03408005 1.57597449 0.00894855 0.09575372]
Intercept:  7.076321160791153
```



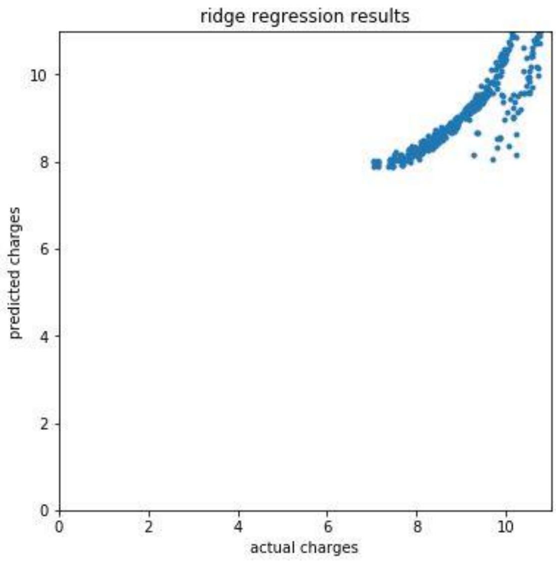
```
#for Polynomial Regression
scores=[]
lr4=LinearRegression()
poly2 = PolynomialFeatures(degree=2)
for train_index,test_index in kf.split(X):
    trainx,testx,trainy,testy=(X.iloc[train_index,:],
                               X.iloc[test_index,:],
                               y[train_index],
                               y[test_index])
    train_x_poly2 = poly2.fit_transform(trainx)
    lr4.fit(train_x_poly2,trainy)
    test_x_poly2 = poly2.fit_transform(testx)
    y_pr2=lr4.predict(test_x_poly2)
    score=r2_score(testy.values,y_pr2)
    scores.append(score)
scores
```

```
[0.8474777101848137, 0.8516935010536314, 0.782415633049339]
```

- Ridge Regression

```
#Ridge regression
from sklearn.linear_model import RidgeCV
alphas=[0.005,0.5,0.1,0.3,1,3,5,10,15,30,80]
ridgeCV=RidgeCV(alphas=alphas,cv=3).fit(X_train,y_train)
y_predictions=ridgeCV.predict(X_test)
print("Mean absolute error: %.2f" % np.mean(np.absolute(y_predictions- y_test)))
print("Residual sum of squares (MSE): %.2f" % np.mean((y_predictions- y_test) ** 2))
print("R2-score: %.2f" % r2_score( y_predictions, y_test) )
```

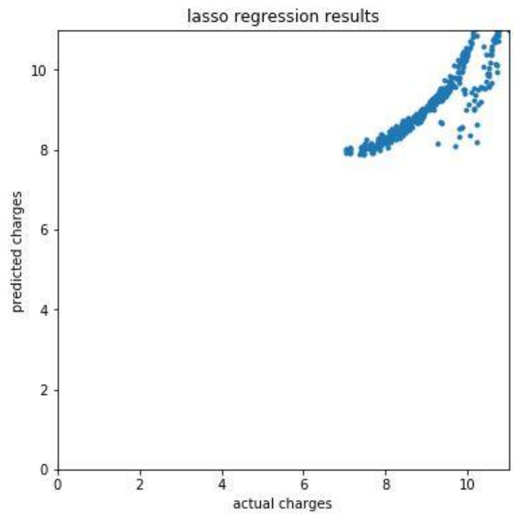
```
Mean absolute error: 0.27
Residual sum of squares (MSE): 0.17
R2-score: 0.75
```



- Lasso Regression

```
#Lasso Regression
from sklearn.linear_model import LassoCV
alphas2=np.array([0.005,0.05,0.1,1,5,20,50,80,100,120,140])
lassoCV=LassoCV(alphas=alphas2,max_iter=5e4,cv=3).fit(X_train,y_train)
y_p=lassoCV.predict(X_test)
print("Mean absolute error: %.2f" % np.mean(np.absolute(y_p- y_test)))
print("Residual sum of squares (MSE): %.2f" % np.mean((y_p- y_test) ** 2))
print("R2-score: %.2f" % r2_score( y_p, y_test) )
```

```
Mean absolute error: 0.27
Residual sum of squares (MSE): 0.17
R2-score: 0.74
```



Accuracy Metrics

Linear Regression:

```
print("Mean absolute error: %.2f" % np.me
print("Residual sum of squares (MSE): %.2
print("R2-score: %.2f" % r2_score( y_pre
```

```
Mean absolute error: 0.27
Residual sum of squares (MSE): 0.17
R2-score: 0.75
```

Polynomial Regression:

```
#Evaluating
test_x_poly = poly.fit_transform(test_x)
y_predict= lr2.predict(test_x_poly)
print(y_predict[0:5],test_y[0:5])
print("Mean absolute error: %.2f" % np.mean(np.absolute(y_predict- test_y)))
print("Residual sum of squares (MSE): %.2f" % np.mean((y_predict- test_y) ** 2))
print("R2-score: %.2f" % r2_score( y_predict, test_y) )
```

```
[8.92142857 8.25197786 8.24146271 8.61847518 9.38720765] [8.92634569 8.1049429  7.96105321 8.40384645 9.93837294]
Mean absolute error: 0.20
Residual sum of squares (MSE): 0.12
R2-score: 0.83
```

Ridge Regression:

```
print("Mean absolute error: %.2f" % np.me
print("Residual sum of squares (MSE): %.2
print("R2-score: %.2f" % r2_score( y_pred
```

```
Mean absolute error: 0.27
Residual sum of squares (MSE): 0.17
R2-score: 0.75
```

Lasso Regression:

```
y_p=lassoCV.predict(X_test)
print("Mean absolute error: %.2f" % np.mean(
print("Residual sum of squares (MSE): %.2f"
print("R2-score: %.2f" % r2_score( y_p, y_te
```

```
Mean absolute error: 0.27
Residual sum of squares (MSE): 0.17
R2-score: 0.74
```

From the Results we can clearly see that r2 score value is highest for the Polynomial Regression model. Also, the mean-squared error value is lowest for Polynomial regression. So, according to the results I feel that the Polynomial Regression model is the best suited model for the given data and attributes.

I do not suggest that this is the only best model. Further studies and revision of the model should take place. The data should also be analysed more carefully and with other hypothesis. It is very much possible that with more exploration and examination of data and its attributes will result in a much better model. Also, adding more features to the data and improving the data set would help in predicting better results.