FINAL PROJECT

Predicting Insurance Premium with given risk factors: Pre-processing and Exploratory data analysis, Feature Engineering, Feature Selection, Model, Pickle, Docker, Flask

COURSE: INFO7390

Advance Data Science & Architecture

PROFESSOR:

Srikanth Krishnamurthy

SUBMITTED BY:

TEAM 9

Amit Pingale - 001898697

Himani Solanki - 001899580

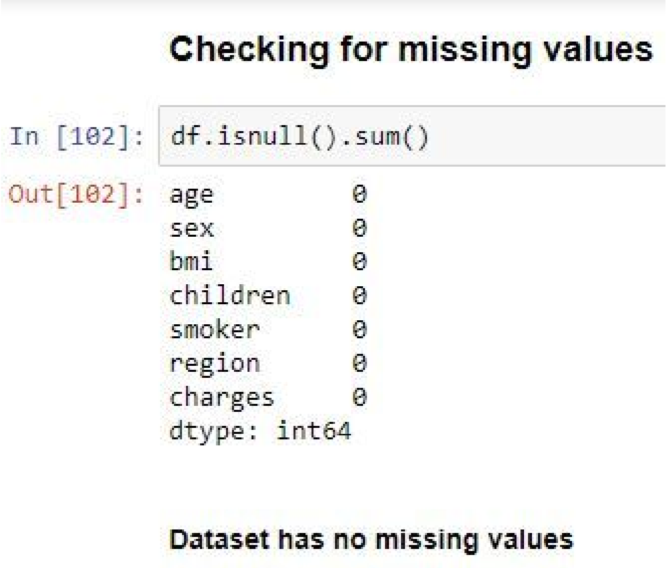
Shubham Patel - 001899476

Objective:

The Report summarizes the design and implementation of machine learning performed on the Insurance dataset. The documentation is divided into 6 parts:

Part 1: Feature Engineering

Step 1:Missing value analysis



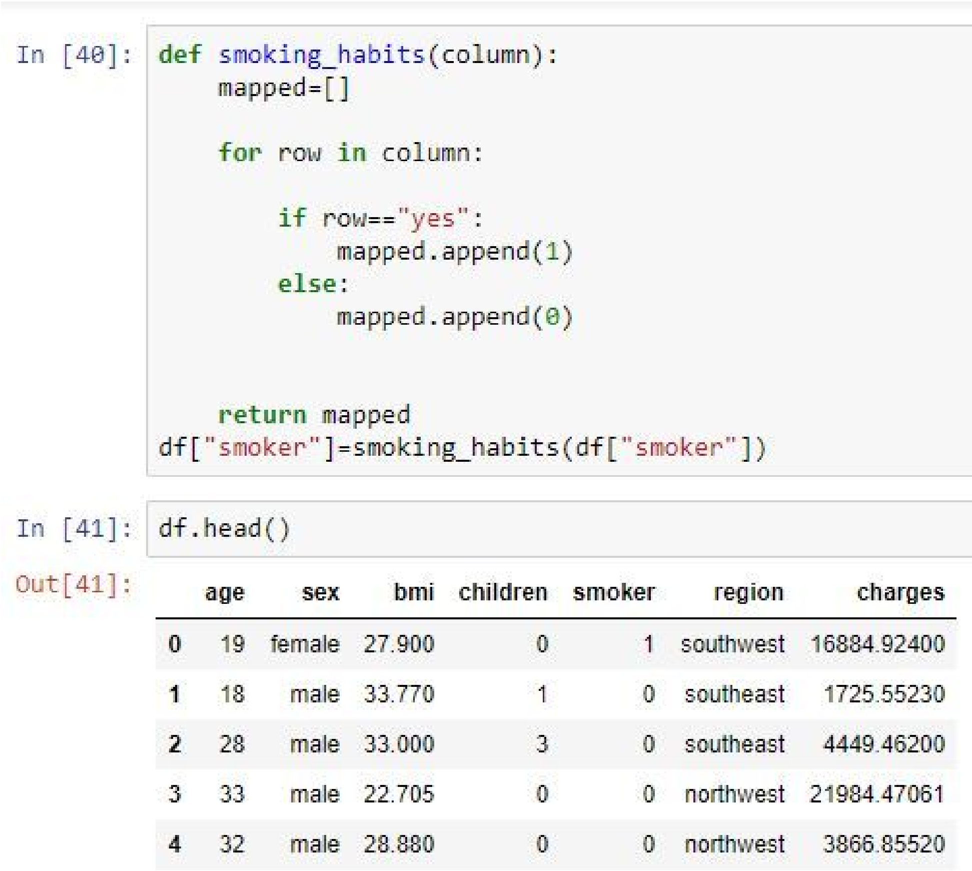
Checking for missing values

```
In [102]: df.isnull().sum()
```

```
Out[102]: age         0
          sex         0
          bmi         0
          children    0
          smoker      0
          region      0
          charges     0
          dtype: int64
```

Dataset has no missing values

Step 2:Converting necessary column to binary value for evaluation

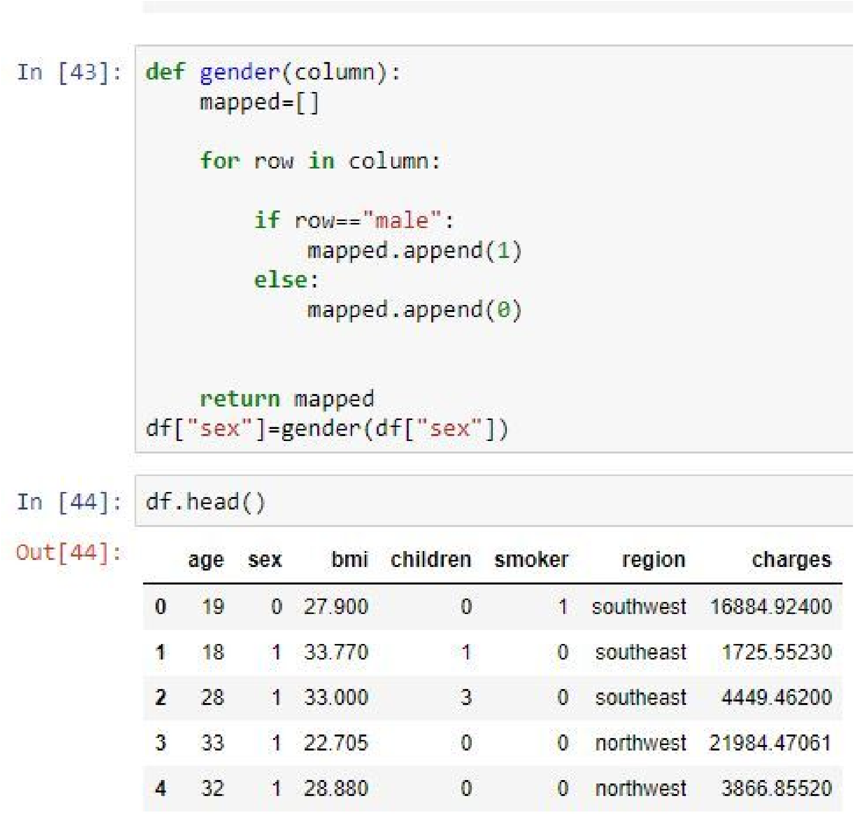


```
In [40]: def smoking_habits(column):
             mapped=[]

             for row in column:

                 if row=="yes":
                     mapped.append(1)
                 else:
                     mapped.append(0)


             return mapped
         df["smoker"]=smoking_habits(df["smoker"])
```

```
In [41]: df.head()
```

Out[41]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	1	southwest	16884.92400
1	18	male	33.770	1	0	southeast	1725.55230
2	28	male	33.000	3	0	southeast	4449.46200
3	33	male	22.705	0	0	northwest	21984.47061
4	32	male	28.880	0	0	northwest	3866.85520



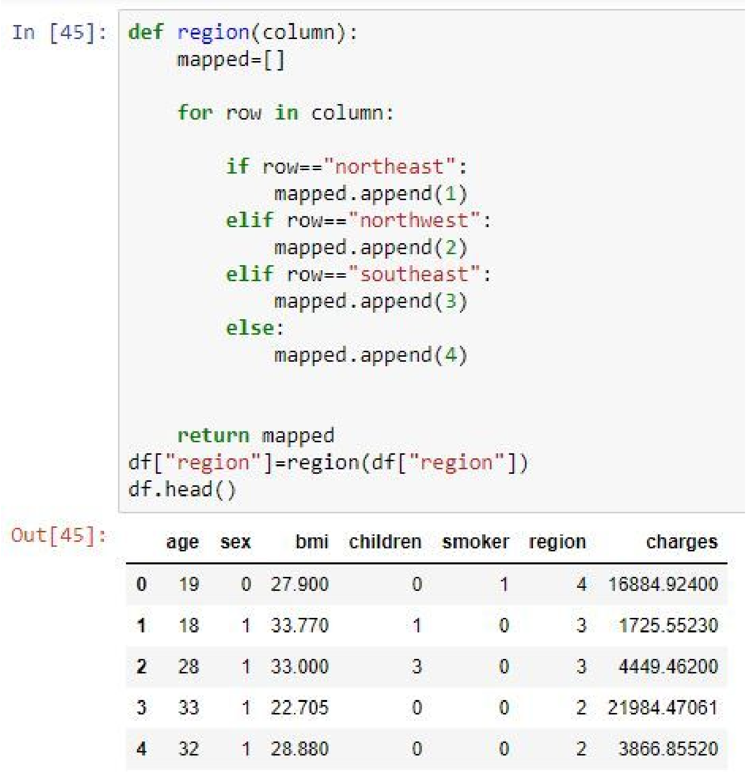
```
In [43]: def gender(column):
             mapped=[]

             for row in column:

                 if row=="male":
                     mapped.append(1)
                 else:
                     mapped.append(0)


             return mapped
         df["sex"]=gender(df["sex"])
```

```
In [44]: df.head()
```

Out[44]:

	age	sex	bmi	children	smoker	region	charges
0	19	0	27.900	0	1	southwest	16884.92400
1	18	1	33.770	1	0	southeast	1725.55230
2	28	1	33.000	3	0	southeast	4449.46200
3	33	1	22.705	0	0	northwest	21984.47061
4	32	1	28.880	0	0	northwest	3866.85520

Step 3: Assigning region indexing



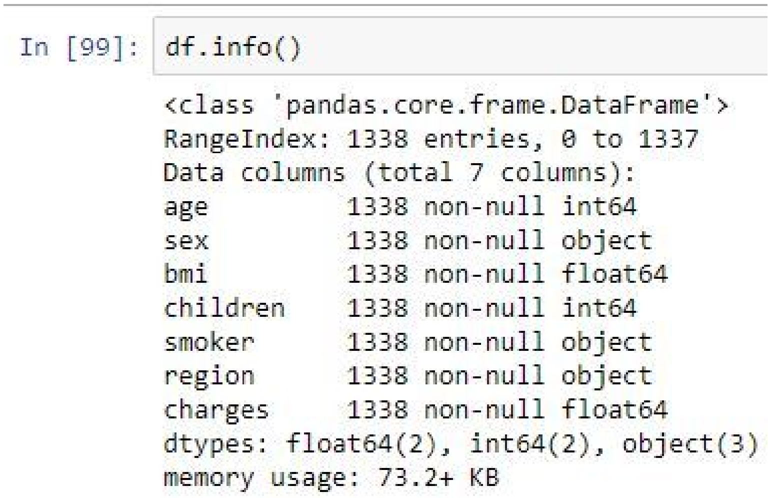
```
In [45]: def region(column):
             mapped=[]

             for row in column:

                 if row=="northeast":
                     mapped.append(1)
                 elif row=="northwest":
                     mapped.append(2)
                 elif row=="southeast":
                     mapped.append(3)
                 else:
                     mapped.append(4)


             return mapped
         df["region"]=region(df["region"])
         df.head()
```

Out[45]:

	age	sex	bmi	children	smoker	region	charges
0	19	0	27.900	0	1	4	16884.92400
1	18	1	33.770	1	0	3	1725.55230
2	28	1	33.000	3	0	3	4449.46200
3	33	1	22.705	0	0	2	21984.47061
4	32	1	28.880	0	0	2	3866.85520

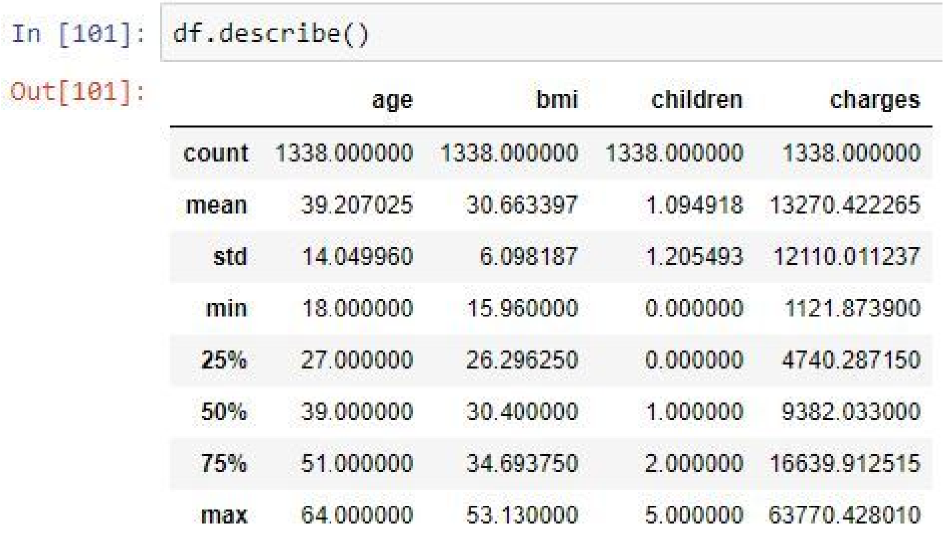
Part 2: Exploratory Data Analysis

Step 1:Data info()



```
In [99]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
age         1338 non-null int64
sex         1338 non-null object
bmi         1338 non-null float64
children    1338 non-null int64
smoker      1338 non-null object
region      1338 non-null object
charges     1338 non-null float64
dtypes: float64(2), int64(2), object(3)
memory usage: 73.2+ KB
```

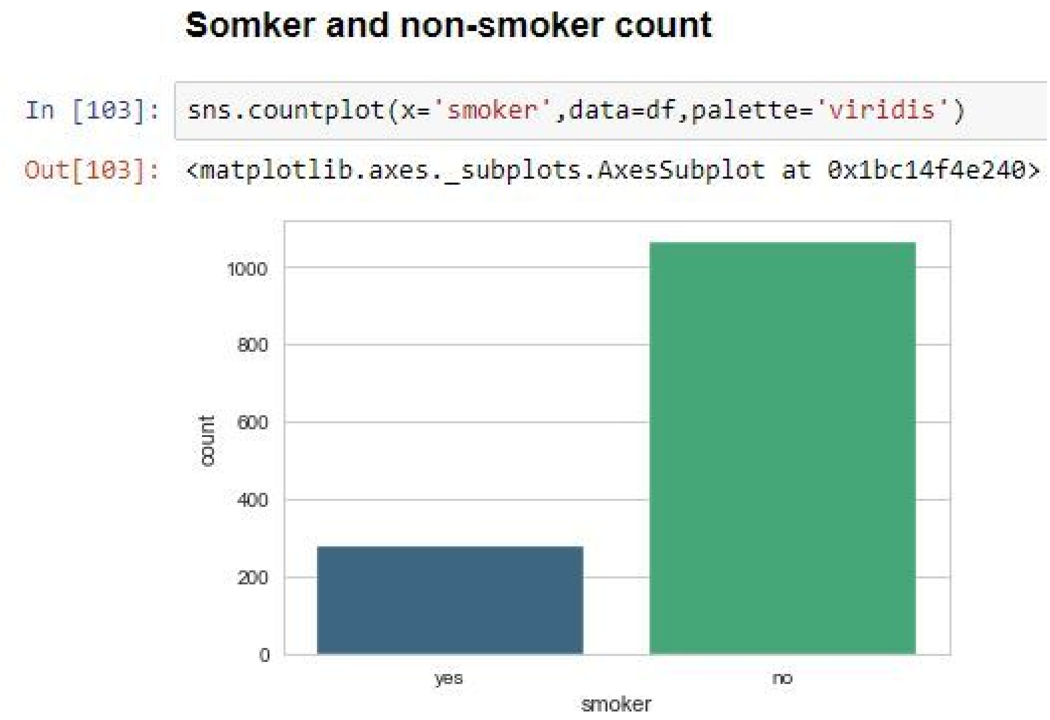
Step 2:Data describe()



```
In [101]: df.describe()
Out[101]:
```

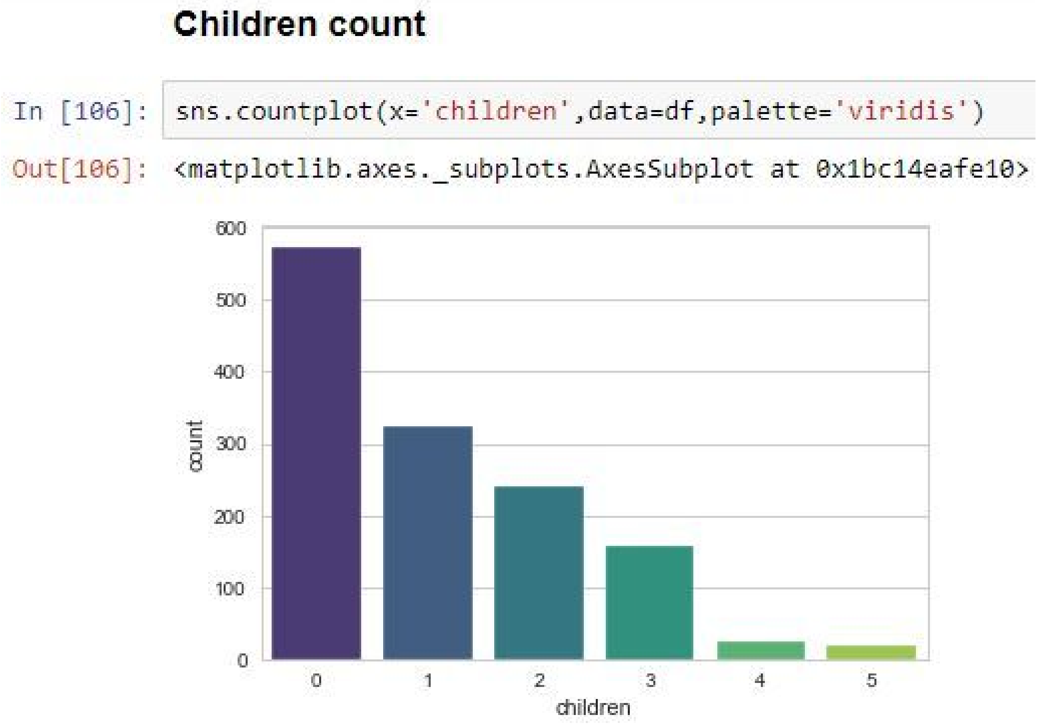
	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	13270.422265
std	14.049960	6.098187	1.205493	12110.011237
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.296250	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	9382.033000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

Step 3:



Somker and non-smoker count

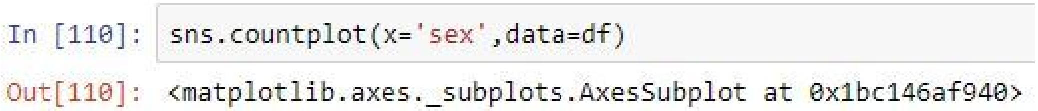
```
In [103]: sns.countplot(x='smoker',data=df,palette='viridis')

Out[103]: <matplotlib.axes._subplots.AxesSubplot at 0x1bc14f4e240>
```



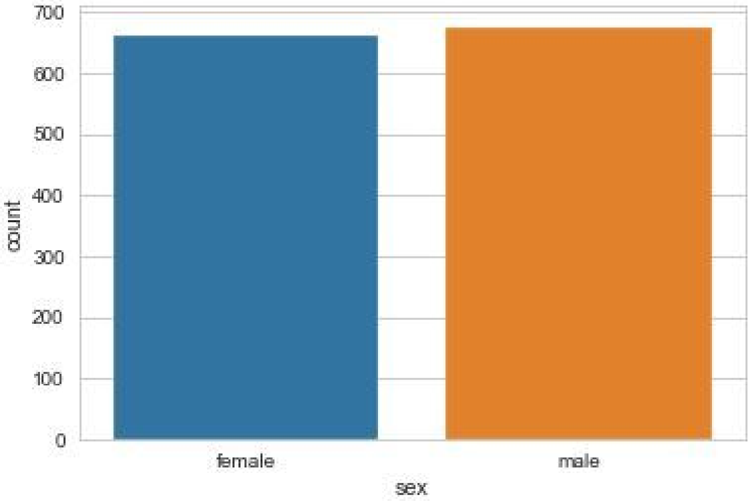
Children count

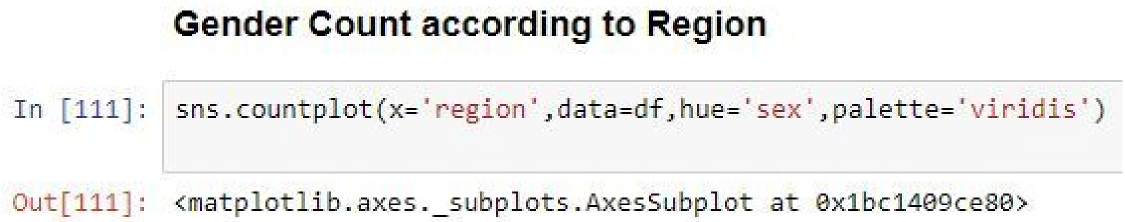
```
In [106]: sns.countplot(x='children',data=df,palette='viridis')

Out[106]: <matplotlib.axes._subplots.AxesSubplot at 0x1bc14eafe10>
```

Step 4: Gender Count



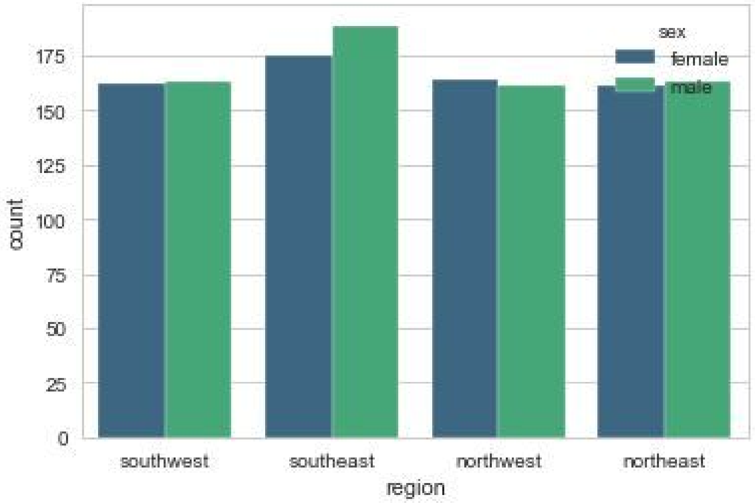
```
In [110]: sns.countplot(x='sex',data=df)

Out[110]: <matplotlib.axes._subplots.AxesSubplot at 0x1bc146af940>
```



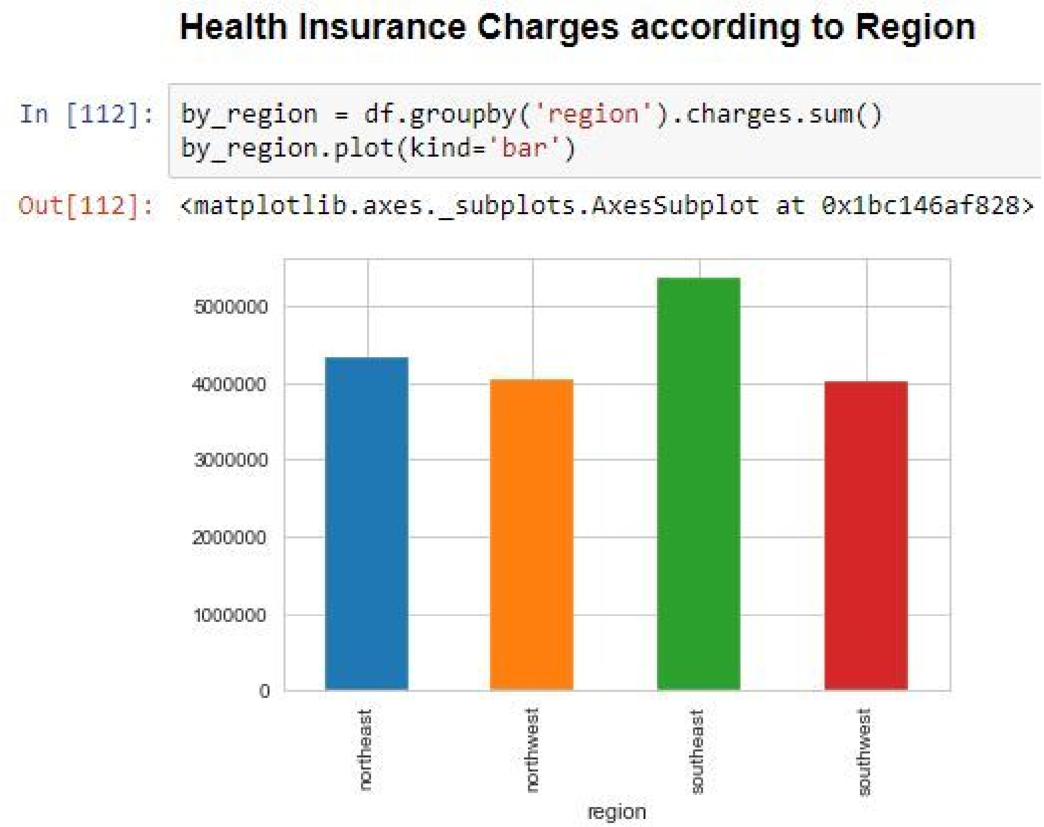


Gender Count according to Region

```
In [111]: sns.countplot(x='region',data=df,hue='sex',palette='viridis')

Out[111]: <matplotlib.axes._subplots.AxesSubplot at 0x1bc1409ce80>
```



Step 5:

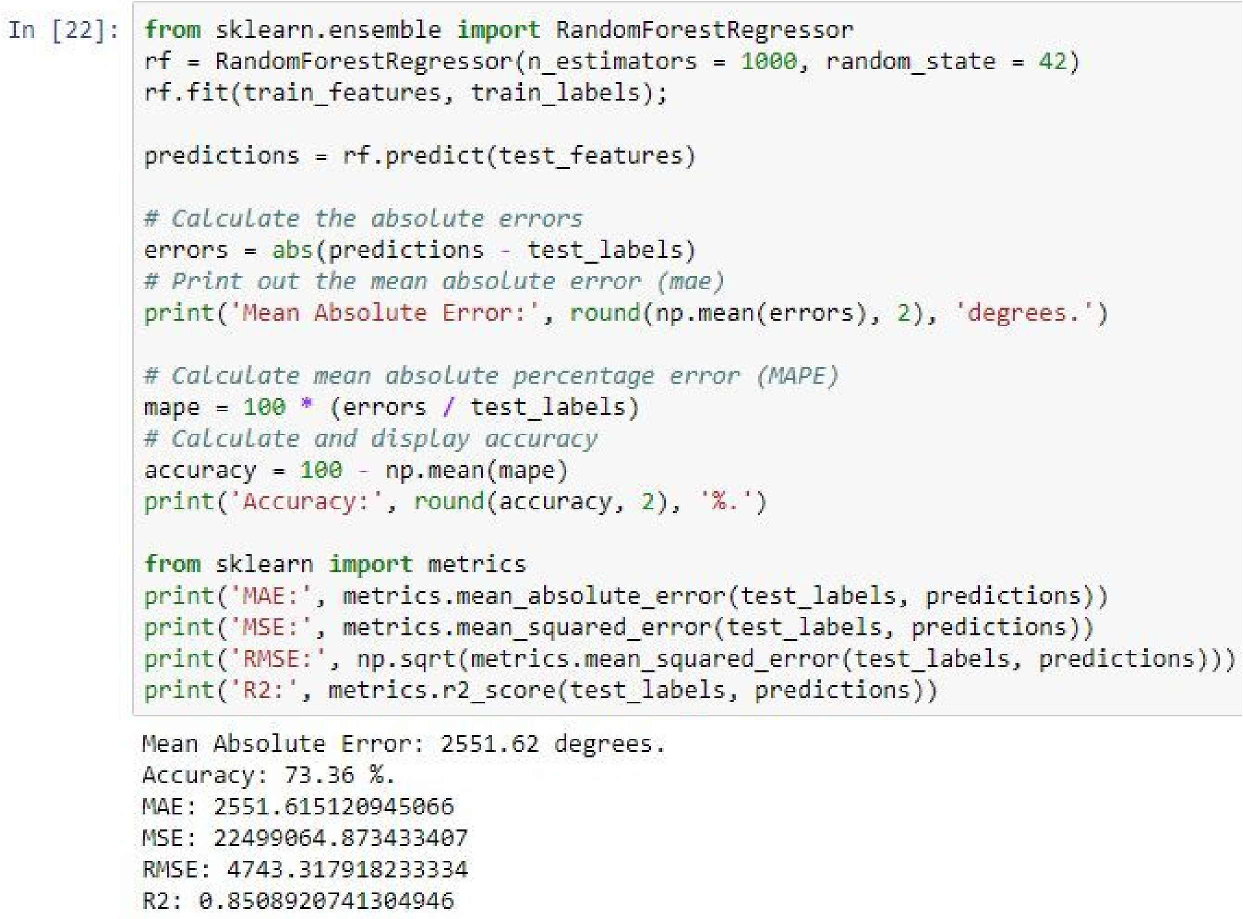


Health Insurance Charges according to Region

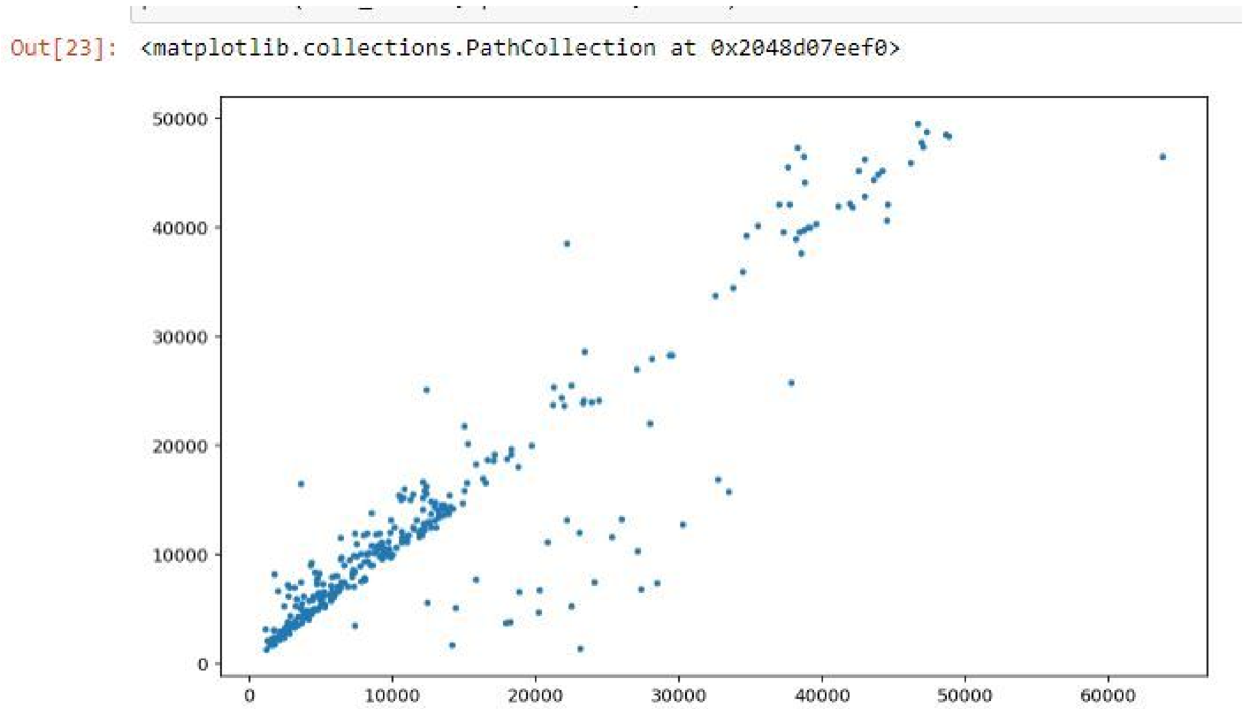
```
In [112]: by_region = df.groupby('region').charges.sum()
          by_region.plot(kind='bar')

Out[112]: <matplotlib.axes._subplots.AxesSubplot at 0x1bc146af828>
```

Part 3: Regression Model

Step 1:Random Forest Regressor

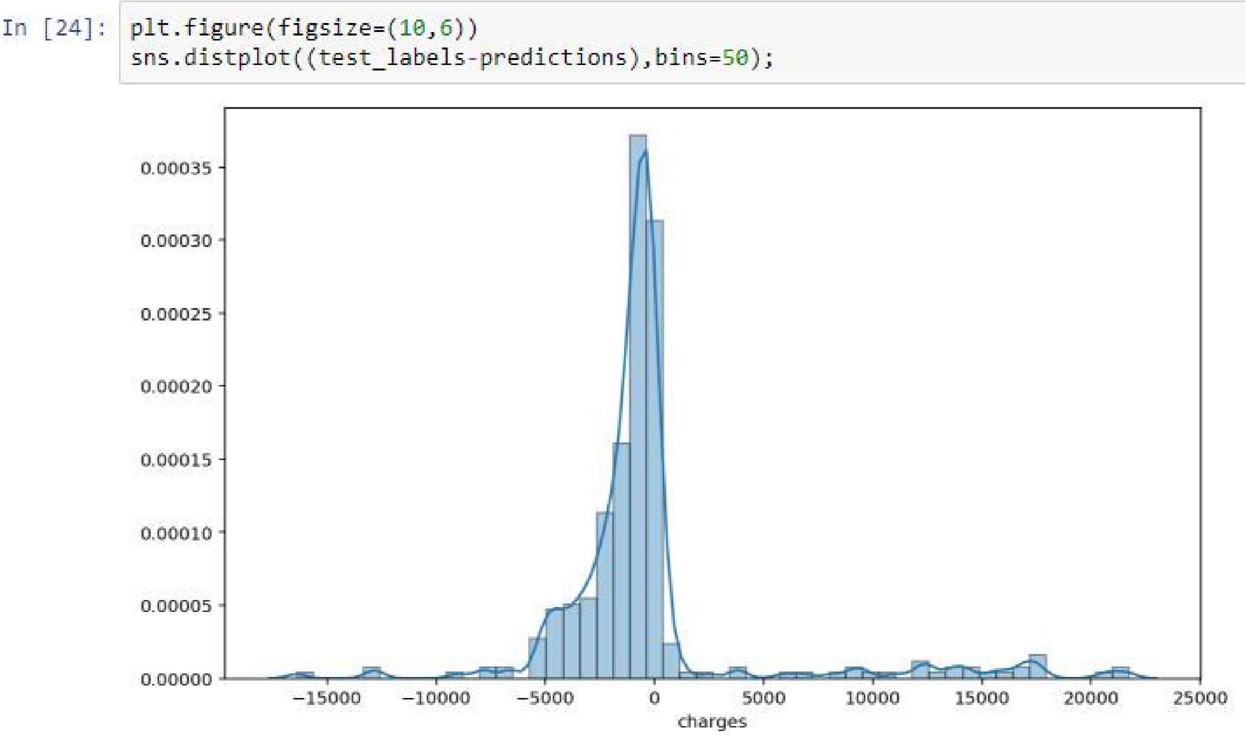


```
In [22]: from sklearn.ensemble import RandomForestRegressor
         rf = RandomForestRegressor(n_estimators = 1000, random_state = 42)
         rf.fit(train_features, train_labels);

         predictions = rf.predict(test_features)

         # Calculate the absolute errors
         errors = abs(predictions - test_labels)
         # Print out the mean absolute error (mae)
         print('Mean Absolute Error:', round(np.mean(errors), 2), 'degrees.')

         # Calculate mean absolute percentage error (MAPE)
         mape = 100 * (errors / test_labels)
         # Calculate and display accuracy
         accuracy = 100 - np.mean(mape)
         print('Accuracy:', round(accuracy, 2), '%.')

         from sklearn import metrics
         print('MAE:', metrics.mean_absolute_error(test_labels, predictions))
         print('MSE:', metrics.mean_squared_error(test_labels, predictions))
         print('RMSE:', np.sqrt(metrics.mean_squared_error(test_labels, predictions)))
         print('R2:', metrics.r2_score(test_labels, predictions))

         Mean Absolute Error: 2551.62 degrees.
         Accuracy: 73.36 %.
         MAE: 2551.615120945066
         MSE: 22499064.873433407
         RMSE: 4743.317918233334
         R2: 0.8508920741304946
```



```
Out[23]: <matplotlib.collections.PathCollection at 0x2048d07eef0>
```

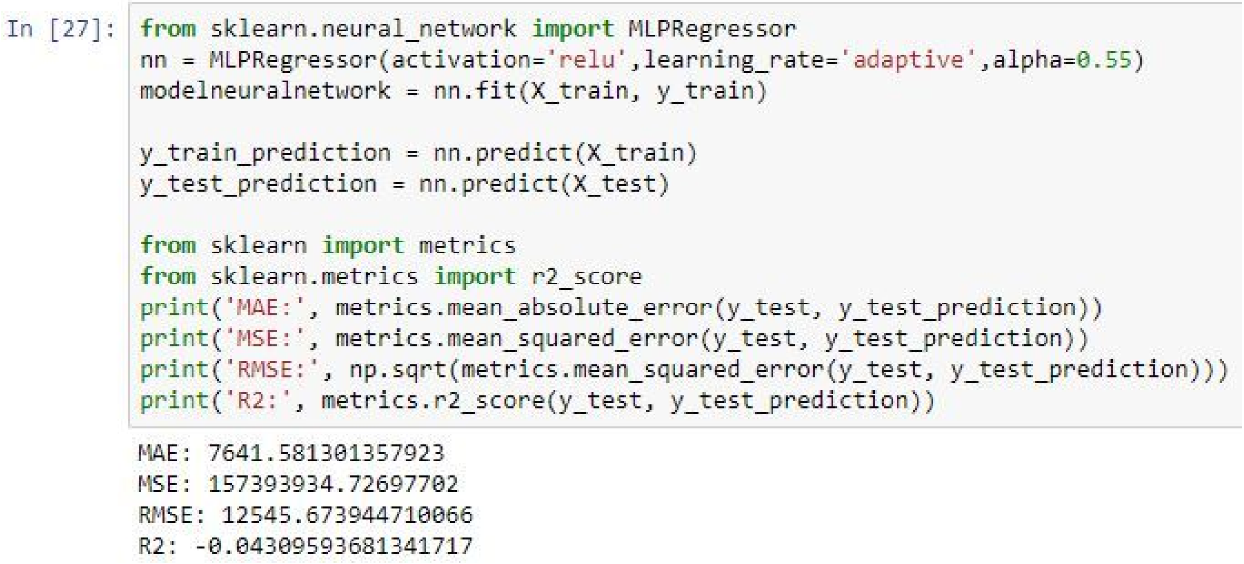
Fig. Scatter Plot for Random Forest Regressor



```
In [24]: plt.figure(figsize=(10,6))
         sns.distplot((test_labels-predictions),bins=50);
```

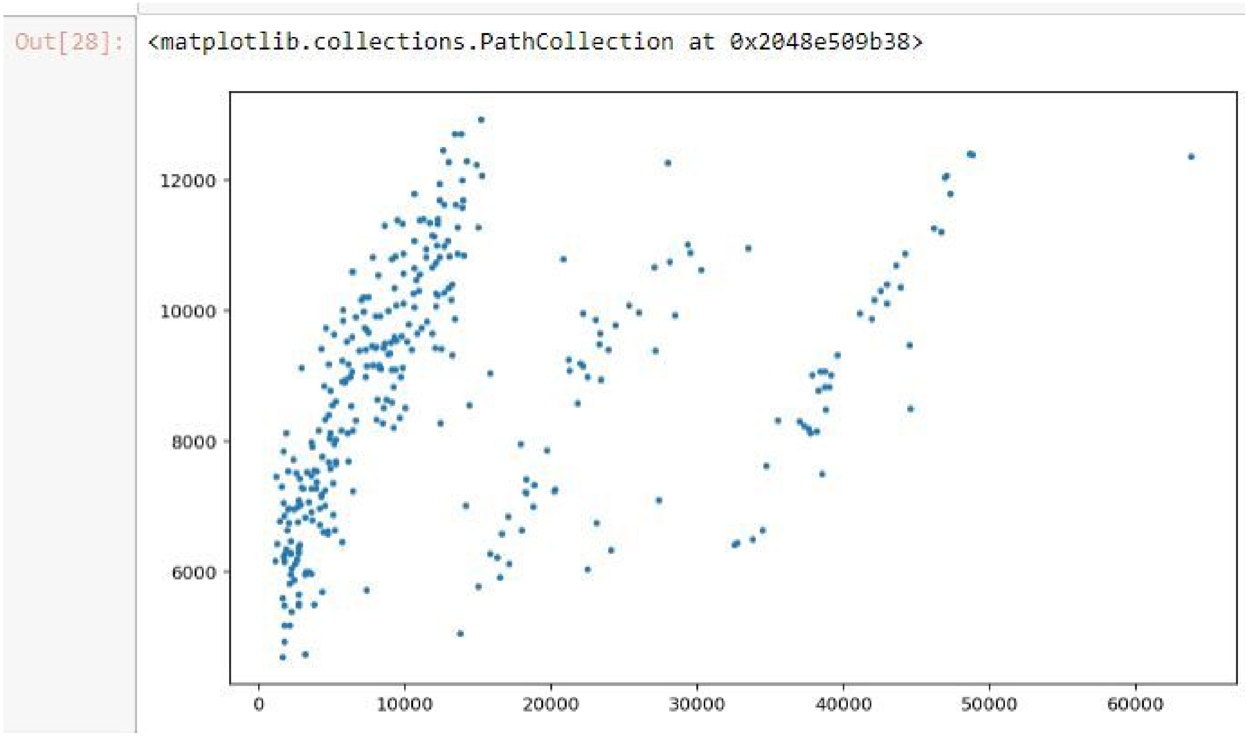
Fig. Shows the error in the Random Forest Regressor

Step 2: Neural Networks



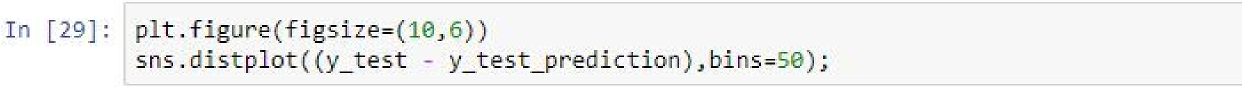
```
In [27]: from sklearn.neural_network import MLPRegressor
         nn = MLPRegressor(activation='relu',learning_rate='adaptive',alpha=0.55)
         modelneuralnetwork = nn.fit(X_train, y_train)

         y_train_prediction = nn.predict(X_train)
         y_test_prediction = nn.predict(X_test)

         from sklearn import metrics
         from sklearn.metrics import r2_score
         print('MAE:', metrics.mean_absolute_error(y_test, y_test_prediction))
         print('MSE:', metrics.mean_squared_error(y_test, y_test_prediction))
         print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_test_prediction)))
         print('R2:', metrics.r2_score(y_test, y_test_prediction))
```

```
MAE: 7641.581301357923
MSE: 157393934.72697702
RMSE: 12545.673944710066
R2: -0.04309593681341717
```



```
Out[28]: <matplotlib.collections.PathCollection at 0x2048e509b38>
```

Fig. Shows the scatter plot for Neural Network



```
In [29]: plt.figure(figsize=(10,6))
         sns.distplot((y_test - y_test_prediction),bins=50);
```

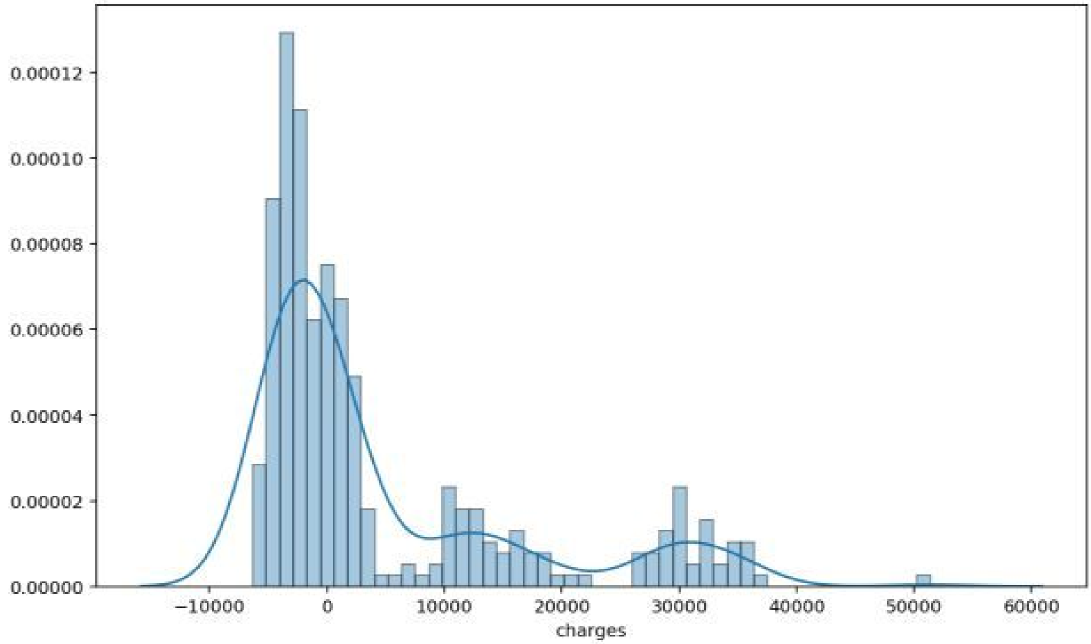
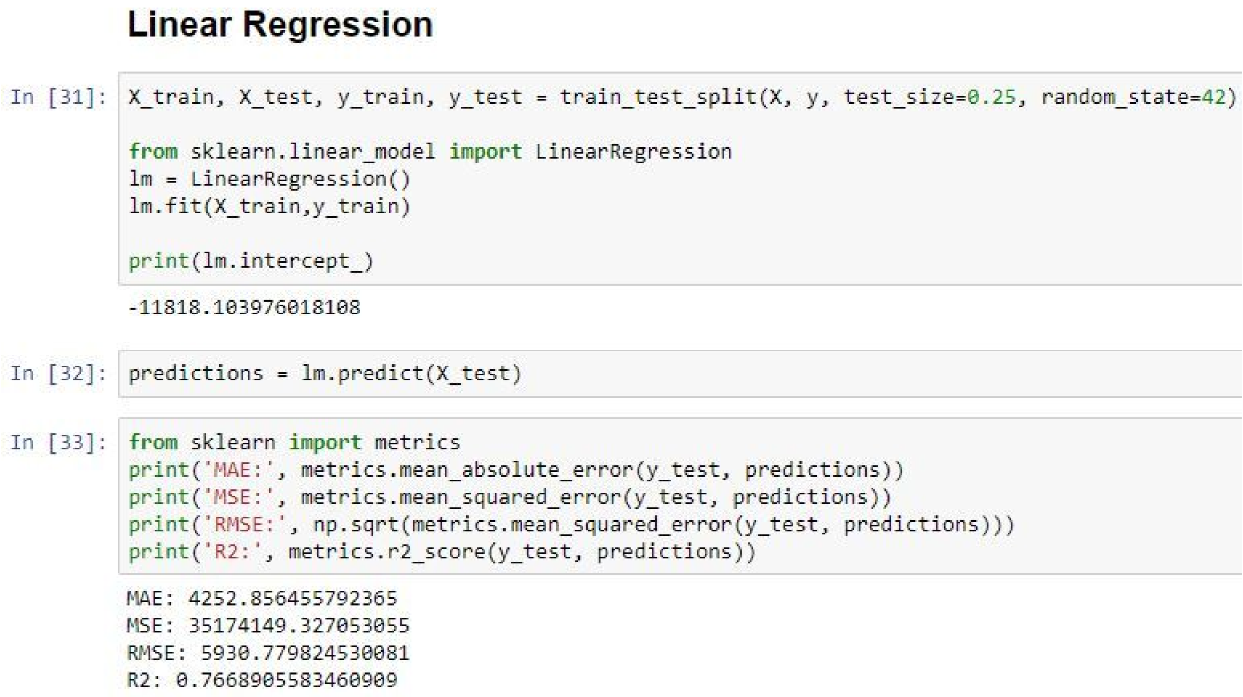


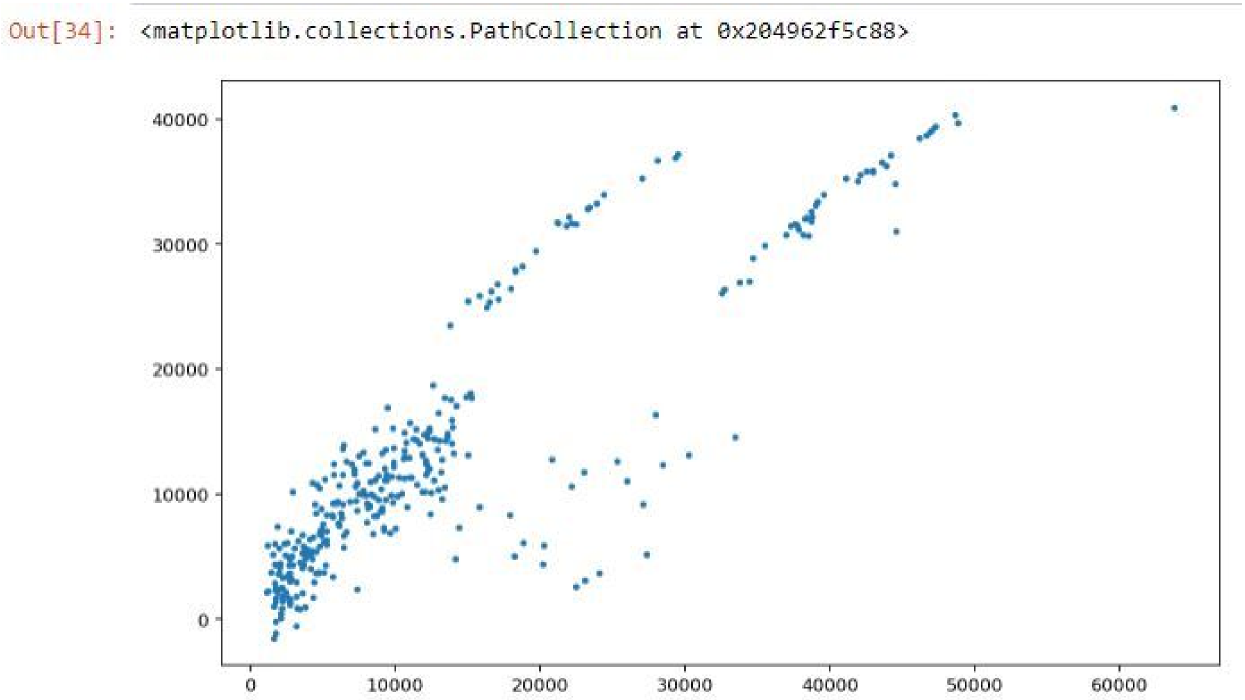
Fig. Shows the error for Neural Networks

Step 3:



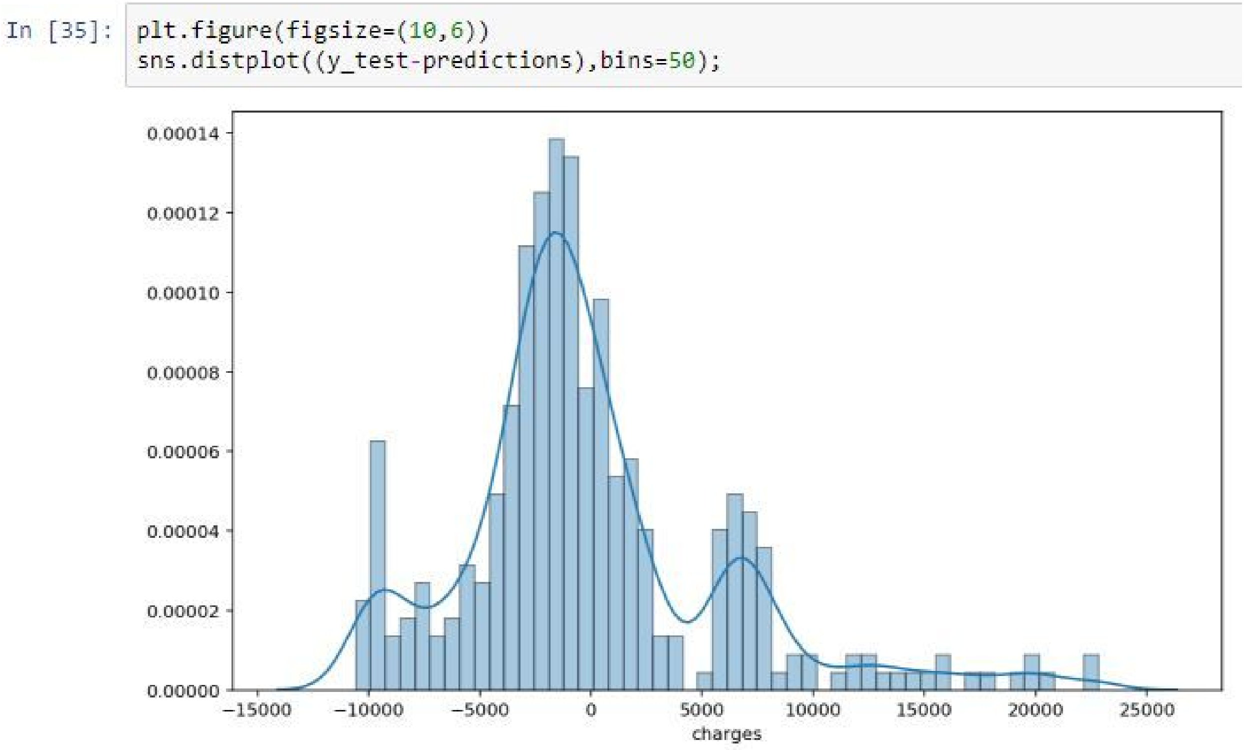
Linear Regression

```
In [31]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

         from sklearn.linear_model import LinearRegression
         lm = LinearRegression()
         lm.fit(X_train,y_train)

         print(lm.intercept_)

         -11818.103976018108

In [32]: predictions = lm.predict(X_test)

In [33]: from sklearn import metrics
         print('MAE:', metrics.mean_absolute_error(y_test, predictions))
         print('MSE:', metrics.mean_squared_error(y_test, predictions))
         print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
         print('R2:', metrics.r2_score(y_test, predictions))

         MAE: 4252.856455792365
         MSE: 35174149.327053055
         RMSE: 5930.779824530081
         R2: 0.7668905583460909
```



```
Out[34]: <matplotlib.collections.PathCollection at 0x204962f5c88>
```

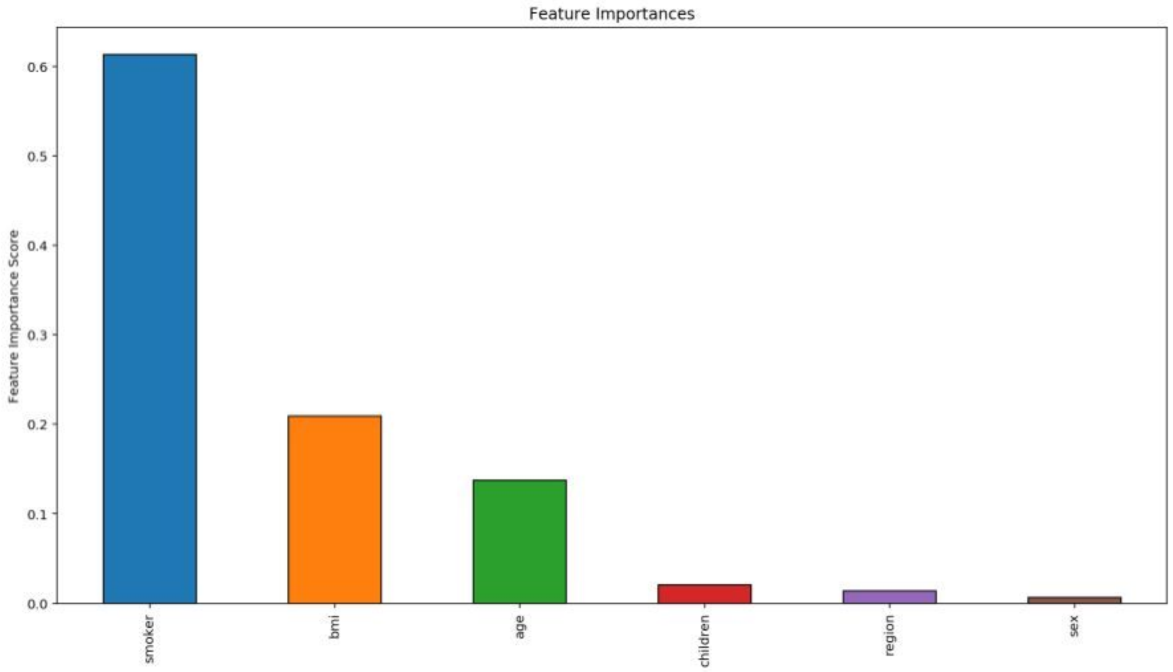
Fig. Showing scatter plot for linear regression



```
In [35]: plt.figure(figsize=(10,6))
         sns.distplot((y_test-predictions),bins=50);
```

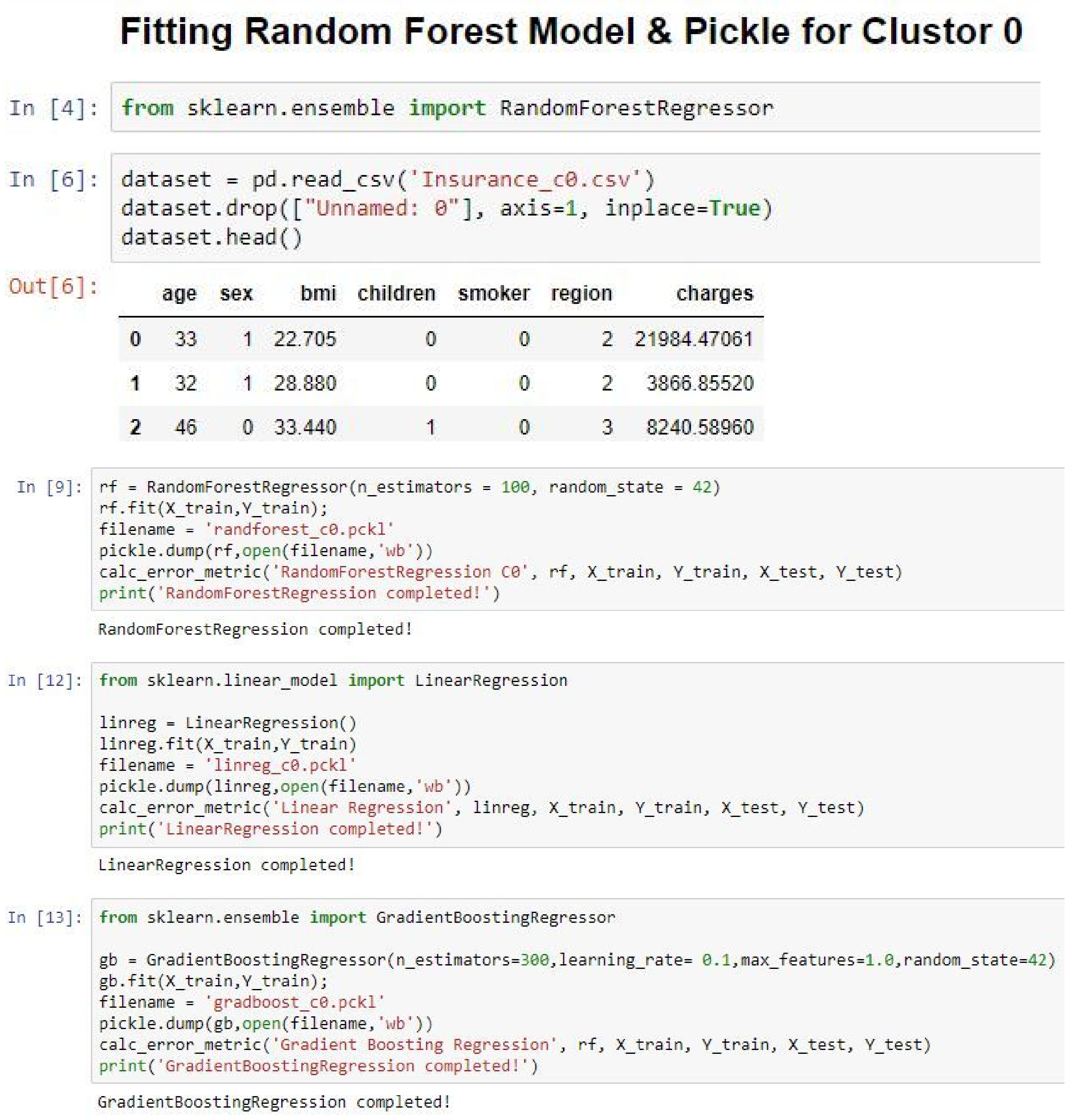
Fig. Shows the error plot for linear regression

Step 4:


Feature Importances
Feature Importance Score
0.0
0.1
0.2
0.3
0.4
0.5
0.6
smoker
bmi
age
children
region
sex

Part 4: Pickle Model

Step 1:



Fitting Random Forest Model & Pickle for Clustor 0

In [4]:
```
from sklearn.ensemble import RandomForestRegressor
```

In [6]:
```
dataset = pd.read_csv('Insurance_c0.csv')
dataset.drop(["Unnamed: 0"], axis=1, inplace=True)
dataset.head()
```

Out[6]:

	age	sex	bmi	children	smoker	region	charges
0	33	1	22.705	0	0	2	21984.47061
1	32	1	28.880	0	0	2	3866.85520
2	46	0	33.440	1	0	3	8240.58960

In [9]:
```
rf = RandomForestRegressor(n_estimators = 100, random_state = 42)
rf.fit(X_train,Y_train);
filename = 'randforest_c0.pckl'
pickle.dump(rf,open(filename,'wb'))
calc_error_metric('RandomForestRegression C0', rf, X_train, Y_train, X_test, Y_test)
print('RandomForestRegression completed!')
```

```
RandomForestRegression completed!
```

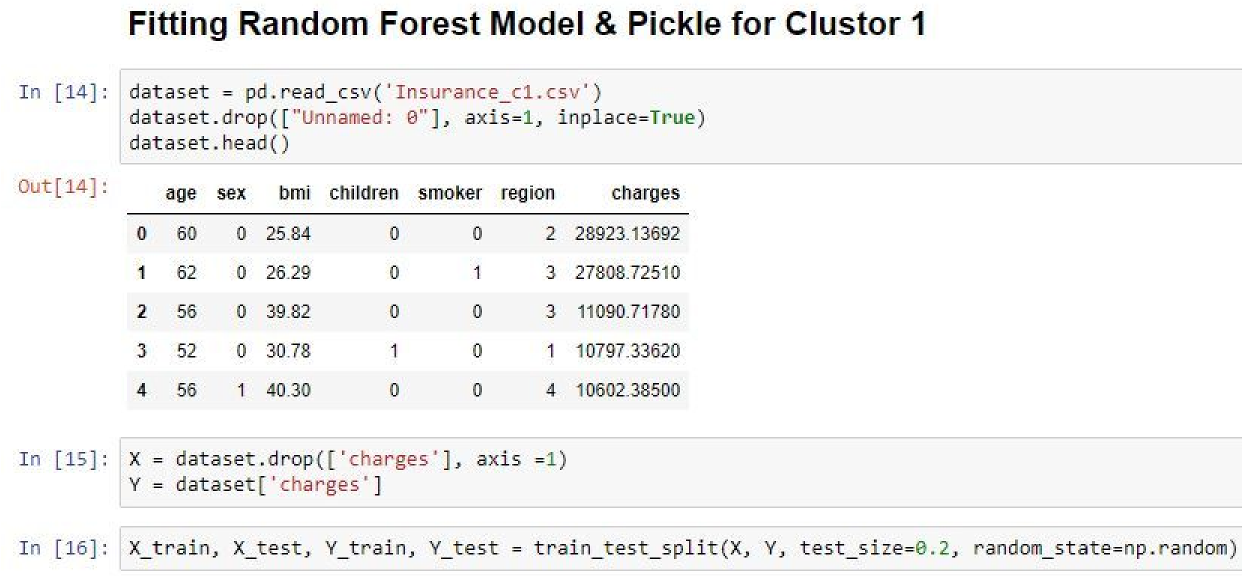
In [12]:
```
from sklearn.linear_model import LinearRegression

linreg = LinearRegression()
linreg.fit(X_train,Y_train)
filename = 'linreg_c0.pckl'
pickle.dump(linreg,open(filename,'wb'))
calc_error_metric('Linear Regression', linreg, X_train, Y_train, X_test, Y_test)
print('LinearRegression completed!')
```

```
LinearRegression completed!
```

In [13]:
```
from sklearn.ensemble import GradientBoostingRegressor

gb = GradientBoostingRegressor(n_estimators=300,learning_rate= 0.1,max_features=1.0,random_state=42)
gb.fit(X_train,Y_train);
filename = 'gradboost_c0.pckl'
pickle.dump(gb,open(filename,'wb'))
calc_error_metric('Gradient Boosting Regression', rf, X_train, Y_train, X_test, Y_test)
print('GradientBoostingRegression completed!')
```

```
GradientBoostingRegression completed!
```

Step 2:



Fitting Random Forest Model & Pickle for Clustor 1

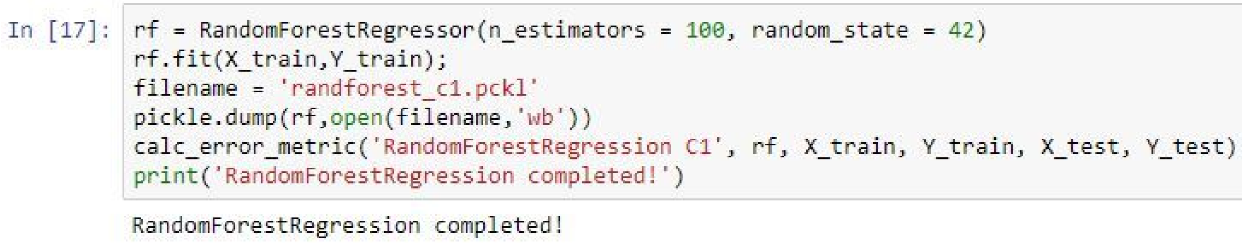
```
In [14]: dataset = pd.read_csv('Insurance_c1.csv')
         dataset.drop(["Unnamed: 0"], axis=1, inplace=True)
         dataset.head()
```

Out[14]:

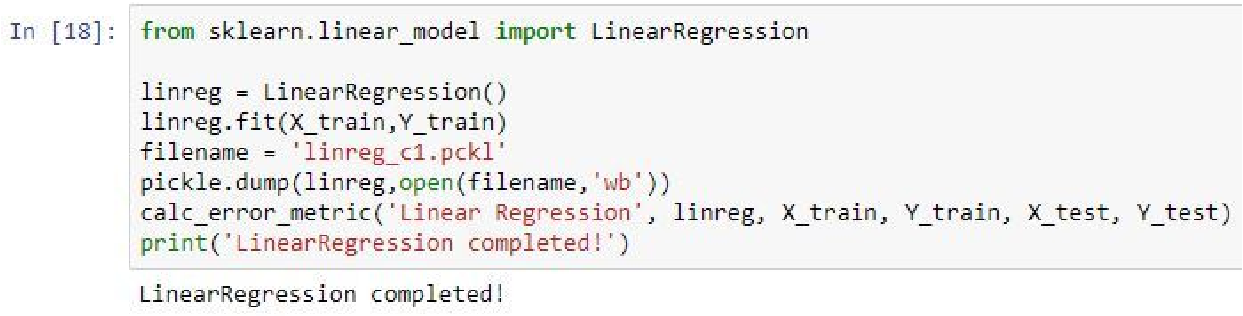
	age	sex	bmi	children	smoker	region	charges
0	60	0	25.84	0	0	2	28923.13692
1	62	0	26.29	0	1	3	27808.72510
2	56	0	39.82	0	0	3	11090.71780
3	52	0	30.78	1	0	1	10797.33620
4	56	1	40.30	0	0	4	10602.38500

```
In [15]: X = dataset.drop(['charges'], axis =1)
         Y = dataset['charges']
```

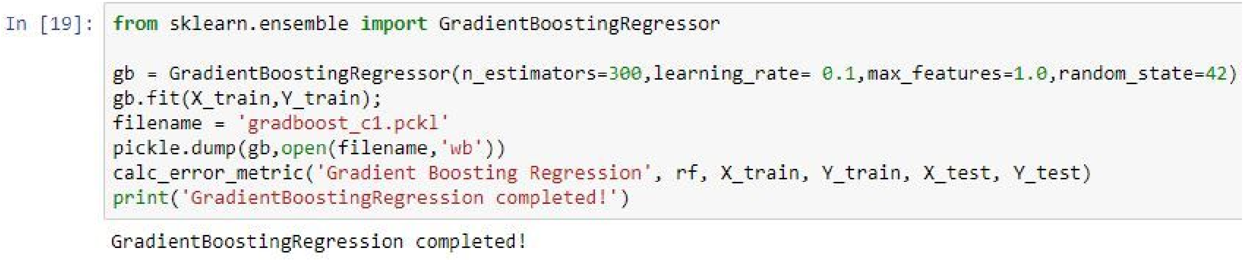
```
In [16]: X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=np.random)
```



```
In [17]: rf = RandomForestRegressor(n_estimators = 100, random_state = 42)
         rf.fit(X_train,Y_train);
         filename = 'randforest_c1.pckl'
         pickle.dump(rf,open(filename,'wb'))
         calc_error_metric('RandomForestRegression C1', rf, X_train, Y_train, X_test, Y_test)
         print('RandomForestRegression completed!')

         RandomForestRegression completed!
```

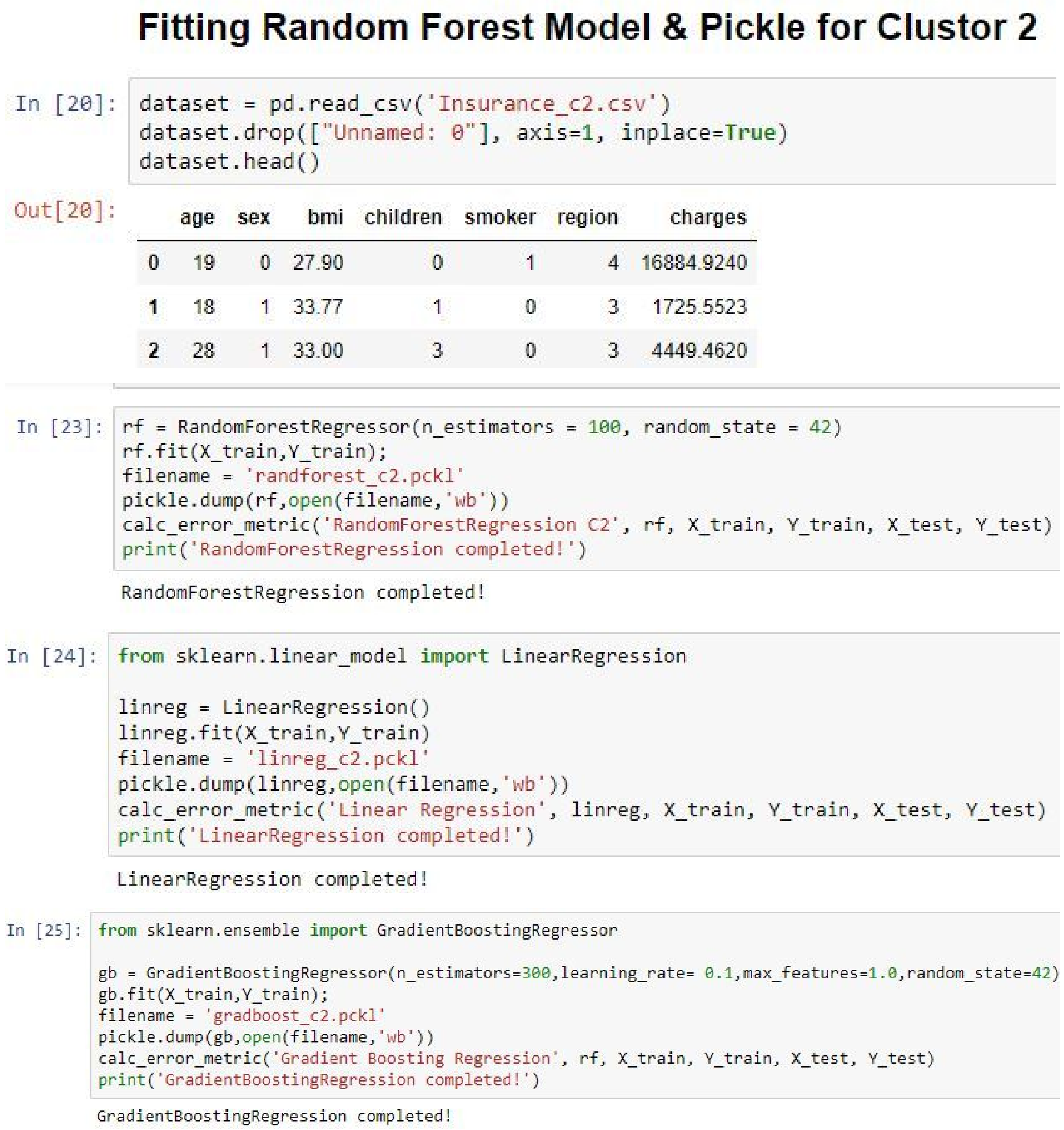


```
In [18]: from sklearn.linear_model import LinearRegression

         linreg = LinearRegression()
         linreg.fit(X_train,Y_train)
         filename = 'linreg_c1.pckl'
         pickle.dump(linreg,open(filename,'wb'))
         calc_error_metric('Linear Regression', linreg, X_train, Y_train, X_test, Y_test)
         print('LinearRegression completed!')

         LinearRegression completed!
```



```
In [19]: from sklearn.ensemble import GradientBoostingRegressor

         gb = GradientBoostingRegressor(n_estimators=300,learning_rate= 0.1,max_features=1.0,random_state=42)
         gb.fit(X_train,Y_train);
         filename = 'gradboost_c1.pckl'
         pickle.dump(gb,open(filename,'wb'))
         calc_error_metric('Gradient Boosting Regression', rf, X_train, Y_train, X_test, Y_test)
         print('GradientBoostingRegression completed!')

         GradientBoostingRegression completed!
```

Step 3:



Fitting Random Forest Model & Pickle for Clustor 2

```
In [20]: dataset = pd.read_csv('Insurance_c2.csv')
         dataset.drop(["Unnamed: 0"], axis=1, inplace=True)
         dataset.head()
```

Out[20]:

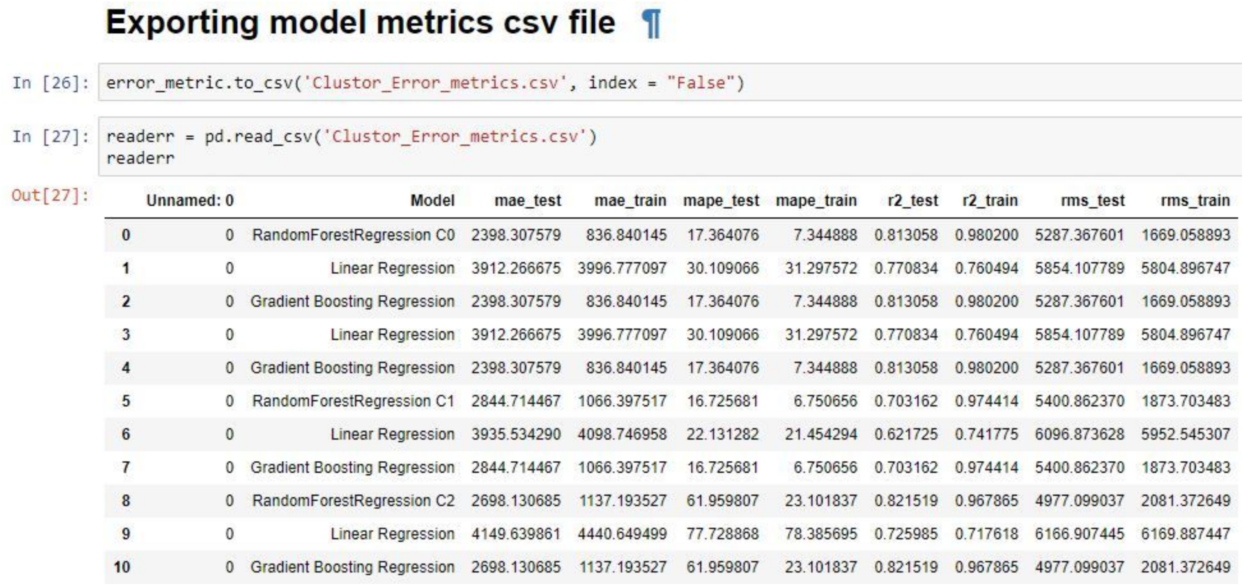
	age	sex	bmi	children	smoker	region	charges
0	19	0	27.90	0	1	4	16884.9240
1	18	1	33.77	1	0	3	1725.5523
2	28	1	33.00	3	0	3	4449.4620

```
In [23]: rf = RandomForestRegressor(n_estimators = 100, random_state = 42)
         rf.fit(X_train,Y_train);
         filename = 'randforest_c2.pckl'
         pickle.dump(rf,open(filename,'wb'))
         calc_error_metric('RandomForestRegression C2', rf, X_train, Y_train, X_test, Y_test)
         print('RandomForestRegression completed!')

         RandomForestRegression completed!
```

```
In [24]: from sklearn.linear_model import LinearRegression

         linreg = LinearRegression()
         linreg.fit(X_train,Y_train)
         filename = 'linreg_c2.pckl'
         pickle.dump(linreg,open(filename,'wb'))
         calc_error_metric('Linear Regression', linreg, X_train, Y_train, X_test, Y_test)
         print('LinearRegression completed!')

         LinearRegression completed!
```

```
In [25]: from sklearn.ensemble import GradientBoostingRegressor

         gb = GradientBoostingRegressor(n_estimators=300,learning_rate= 0.1,max_features=1.0,random_state=42)
         gb.fit(X_train,Y_train);
         filename = 'gradboost_c2.pckl'
         pickle.dump(gb,open(filename,'wb'))
         calc_error_metric('Gradient Boosting Regression', rf, X_train, Y_train, X_test, Y_test)
         print('GradientBoostingRegression completed!')

         GradientBoostingRegression completed!
```

Step 4:



Exporting model metrics csv file ¶

```
In [26]: error_metric.to_csv('Clustor_Error_metrics.csv', index = "False")
```

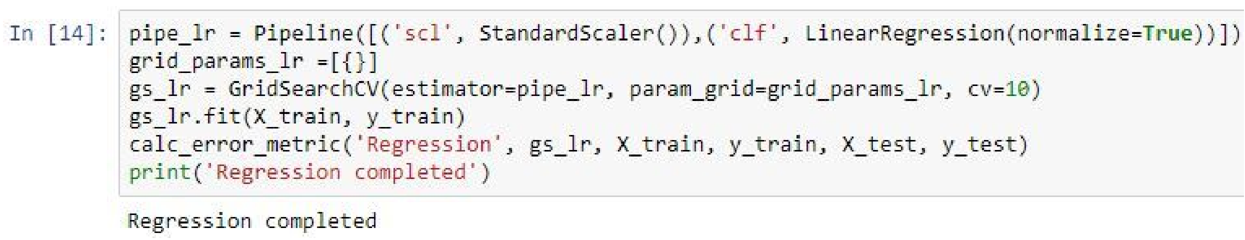
```
In [27]: readerr = pd.read_csv('Clustor_Error_metrics.csv')
         readerr
```

Out[27]:

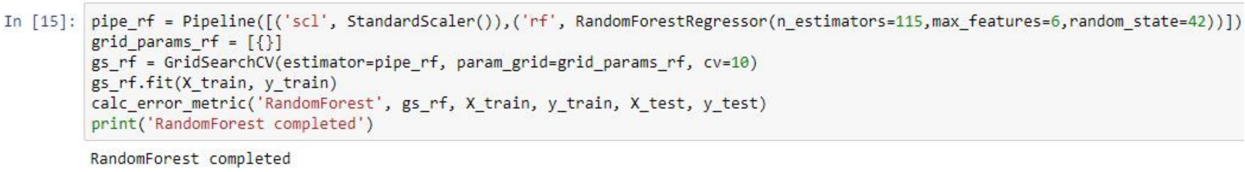
	Unnamed: 0	Model	mae_test	mae_train	mape_test	mape_train	r2_test	r2_train	rms_test	rms_train
0	0	RandomForestRegression C0	2398.307579	836.840145	17.364076	7.344888	0.813058	0.980200	5287.367601	1669.058893
1	0	Linear Regression	3912.266675	3996.777097	30.109066	31.297572	0.770834	0.760494	5854.107789	5804.896747
2	0	Gradient Boosting Regression	2398.307579	836.840145	17.364076	7.344888	0.813058	0.980200	5287.367601	1669.058893
3	0	Linear Regression	3912.266675	3996.777097	30.109066	31.297572	0.770834	0.760494	5854.107789	5804.896747
4	0	Gradient Boosting Regression	2398.307579	836.840145	17.364076	7.344888	0.813058	0.980200	5287.367601	1669.058893
5	0	RandomForestRegression C1	2844.714467	1066.397517	16.725681	6.750656	0.703162	0.974414	5400.862370	1873.703483
6	0	Linear Regression	3935.534290	4098.746958	22.131282	21.454294	0.621725	0.741775	6096.873628	5952.545307
7	0	Gradient Boosting Regression	2844.714467	1066.397517	16.725681	6.750656	0.703162	0.974414	5400.862370	1873.703483
8	0	RandomForestRegression C2	2698.130685	1137.193527	61.959807	23.101837	0.821519	0.967865	4977.099037	2081.372649
9	0	Linear Regression	4149.639861	4440.649499	77.728868	78.385695	0.725985	0.717618	6166.907445	6169.887447
10	0	Gradient Boosting Regression	2698.130685	1137.193527	61.959807	23.101837	0.821519	0.967865	4977.099037	2081.372649

Part 5: Creating the pipeline

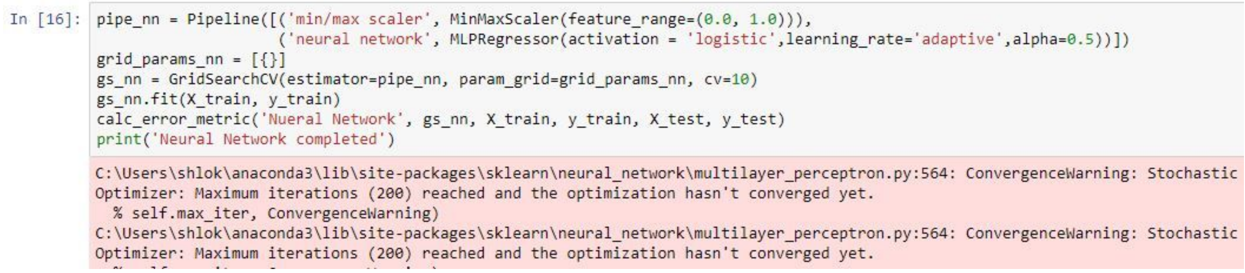
Step 1:Creating the pipeline



```
In [14]: pipe_lr = Pipeline([('scl', StandardScaler()),('clf', LinearRegression(normalize=True))])
         grid_params_lr =[{}]
         gs_lr = GridSearchCV(estimator=pipe_lr, param_grid=grid_params_lr, cv=10)
         gs_lr.fit(X_train, y_train)
         calc_error_metric('Regression', gs_lr, X_train, y_train, X_test, y_test)
         print('Regression completed')

         Regression completed
```

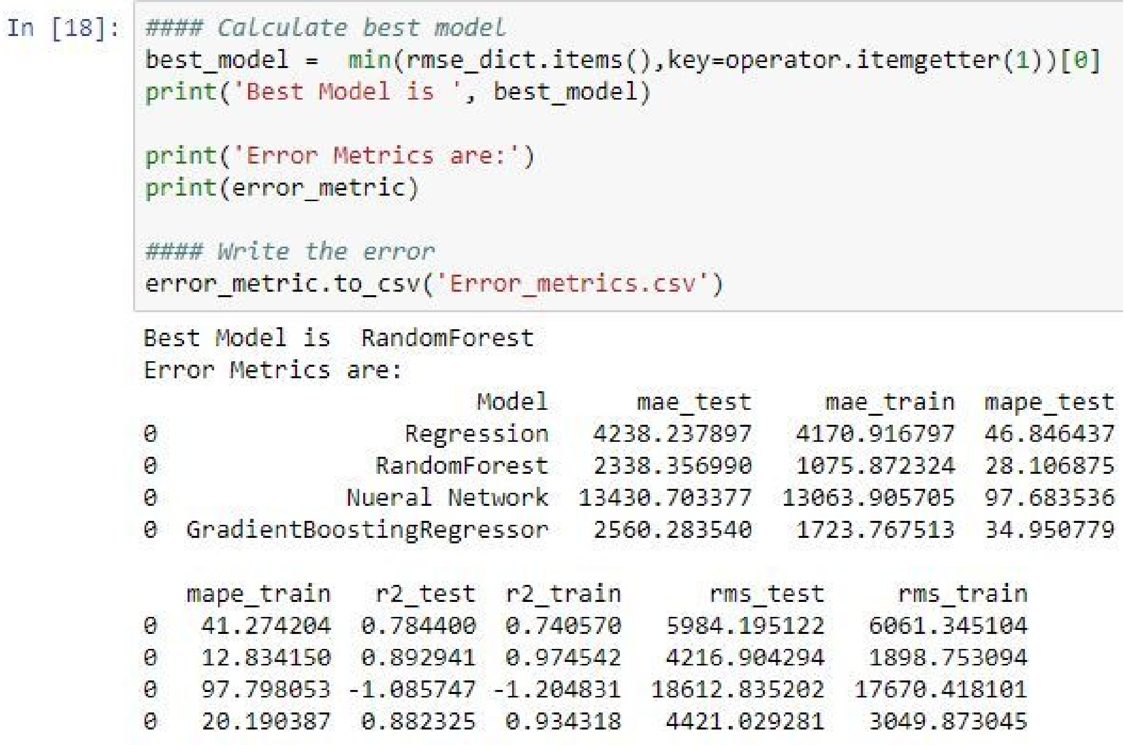


```
In [15]: pipe_rf = Pipeline([('scl', StandardScaler()),('rf', RandomForestRegressor(n_estimators=115,max_features=6,random_state=42))])
         grid_params_rf = [{}]
         gs_rf = GridSearchCV(estimator=pipe_rf, param_grid=grid_params_rf, cv=10)
         gs_rf.fit(X_train, y_train)
         calc_error_metric('RandomForest', gs_rf, X_train, y_train, X_test, y_test)
         print('RandomForest completed')

         RandomForest completed
```



```
In [16]: pipe_nn = Pipeline([('min/max scaler', MinMaxScaler(feature_range=(0.0, 1.0))),
                             ('neural network', MLPRegressor(activation = 'logistic',learning_rate='adaptive',alpha=0.5))])
         grid_params_nn = [{}]
         gs_nn = GridSearchCV(estimator=pipe_nn, param_grid=grid_params_nn, cv=10)
         gs_nn.fit(X_train, y_train)
         calc_error_metric('Nueral Network', gs_nn, X_train, y_train, X_test, y_test)
         print('Neural Network completed')

         C:\Users\shlok\anaconda3\lib\site-packages\sklearn\neural_network\multilayer_perceptron.py:564: ConvergenceWarning: Stochastic
         Optimizer: Maximum iterations (200) reached and the optimization hasn't converged yet.
           % self.max_iter, ConvergenceWarning)
         C:\Users\shlok\anaconda3\lib\site-packages\sklearn\neural_network\multilayer_perceptron.py:564: ConvergenceWarning: Stochastic
         Optimizer: Maximum iterations (200) reached and the optimization hasn't converged yet.
```

Step 2: Calculate best model



```
In [18]: #### Calculate best model
         best_model =  min(rmse_dict.items(),key=operator.itemgetter(1))[0]
         print('Best Model is ', best_model)

         print('Error Metrics are:')
         print(error_metric)

         #### Write the error
         error_metric.to_csv('Error_metrics.csv')

         Best Model is  RandomForest
         Error Metrics are:
                                Model      mae_test     mae_train  mape_test
         0                 Regression   4238.237897   4170.916797  46.846437
         0               RandomForest   2338.356990   1075.872324  28.106875
         0             Nueral Network  13430.703377  13063.905705  97.683536
         0  GradientBoostingRegressor   2560.283540   1723.767513  34.950779

            mape_train   r2_test  r2_train     rms_test     rms_train
         0   41.274204  0.784400  0.740570  5984.195122   6061.345104
         0   12.834150  0.892941  0.974542  4216.904294   1898.753094
         0   97.798053 -1.085747 -1.204831 18612.835202  17670.418101
         0   20.190387  0.882325  0.934318  4421.029281   3049.873045
```

References:

https://www.nhlbi.nih.gov/health/educational/lose_wt/BMI/bmicalc.htm
https://www.kaggle.com/mirichoi0218/insurance