ExploratoryDataAnalysis_Project.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share S

+ Code + Text

RAM Disk

```
[1] !pip install plotly
    import pandas as pd
    import numpy as  np
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: plotly in /usr/local/lib/python3.9/dist-packages (5.13.1)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.9/dist-packages (from plotly) (8.2.2)
```

Creating Dataframe from csv file

This data set is about the insurance prices leveraged for different people depending on many factors

```
[2] df = pd.read_csv("insurance.csv")
    df
```

	index	age	sex	bmi	children	smoker	region	charges
0	0	19	female	27.900	0	yes	southwest	16884.92400
1	1	18	male	33.770	1	no	southeast	1725.55230
2	2	28	male	33.000	3	no	southeast	4449.46200
3	3	33	male	22.705	0	no	northwest	21984.47061
4	4	32	male	28.880	0	no	northwest	3866.85520
...	...	...	...	...	...	...	...	...
1333	1333	50	male	30.970	3	no	northwest	10600.54830
1334	1334	18	female	31.920	0	no	northeast	2205.98080
1335	1335	18	female	36.850	0	no	southeast	1629.83350
1336	1336	21	female	25.800	0	no	southwest	2007.94500
1337	1337	61	female	29.070	0	yes	northwest	29141.36030

1338 rows × 8 columns

```
[3] df.head(10)
```

	index	age	sex	bmi	children	smoker	region	charges
0	0	19	female	27.900	0	yes	southwest	16884.92400
1	1	18	male	33.770	1	no	southeast	1725.55230
2	2	28	male	33.000	3	no	southeast	4449.46200
3	3	33	male	22.705	0	no	northwest	21984.47061
4	4	32	male	28.880	0	no	northwest	3866.85520
5	5	31	female	25.740	0	no	southeast	3756.62160
6	6	46	female	33.440	1	no	southeast	8240.58960
7	7	37	female	27.740	3	no	northwest	7281.50560
8	8	37	male	29.830	2	no	northeast	6406.41070
9	9	60	female	25.840	0	no	northwest	28923.13692

Data Preprocessing

Drop the null values

```
[4] df.dropna(inplace=True)
    df
```

	index	age	sex	bmi	children	smoker	region	charges
0	0	19	female	27.900	0	yes	southwest	16884.92400
1	1	18	male	33.770	1	no	southeast	1725.55230
2	2	28	male	33.000	3	no	southeast	4449.46200
3	3	33	male	22.705	0	no	northwest	21984.47061
4	4	32	male	28.880	0	no	northwest	3866.85520
...	...	...	...	...	...	...	...	...
1333	1333	50	male	30.970	3	no	northwest	10600.54830
1334	1334	18	female	31.920	0	no	northeast	2205.98080
1335	1335	18	female	36.850	0	no	southeast	1629.83350
1336	1336	21	female	25.800	0	no	southwest	2007.94500
1337	1337	61	female	29.070	0	yes	northwest	29141.36030

1338 rows × 8 columns

[5]
```
df.isnull().sum()
```

```
index       0
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

[6]
```
df.dtypes
```

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

[7]
```
df.size
```

```
10704
```

[8]
```
df.sample()
```

	index	age	sex	bmi	children	smoker	region	charges
1294	1294	58	male	25.175	0	no	northeast	11931.12525

Returns number of unique elements in dataframe over rows or columns

[9]
```
df.nunique()
```

```
index       1338
age           47
sex            2
bmi          548
children       6
smoker         2
region         4
charges     1337
dtype: int64
```

Count duplicate values

[10]
```
duplicates = df.duplicated()  # Returns the series denoting duplicated rows
duplicates.sum()
```

```
0
```

Descriptive Statistics

[11]
```
df.shape
```

```
(1338, 8)
```

[12]
```
df.columns
```

```
Index(['index', 'age', 'sex', 'bmi', 'children', 'smoker', 'region',
       'charges'],
      dtype='object')
```

[13]
```
df.describe()
```

	index	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000
mean	668.500000	39.207025	30.663397	1.094918	13270.422265
std	386.391641	14.049960	6.098187	1.205493	12110.011237
min	0.000000	18.000000	15.960000	0.000000	1121.873900
25%	334.250000	27.000000	26.296250	0.000000	4740.287150

50%	668.500000	39.000000	30.400000	1.000000	9382.033000
75%	1002.750000	51.000000	34.693750	2.000000	16639.912515
max	1337.000000	64.000000	53.130000	5.000000	63770.428010

[14] df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 8 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   index     1338 non-null   int64  
 1   age       1338 non-null   int64  
 2   sex       1338 non-null   object 
 3   bmi       1338 non-null   float64
 4   children  1338 non-null   int64  
 5   smoker    1338 non-null   object 
 6   region    1338 non-null   object 
 7   charges   1338 non-null   float64
dtypes: float64(2), int64(3), object(3)
memory usage: 83.8+ KB
```

To get the unique values from a column

[15] len(df['region'].unique())

```
4
```

To count the frequency of unique values in dataframe

[16] df.value_counts()

```
index  age  sex     bmi     children  smoker  region     charges    
0      19   female  27.900  0         yes     southwest  16884.92400    1
898    18   female  40.260  0         no      southeast  1634.57340     1
896    43   female  20.045  2         yes     northeast  19798.05455    1
895    61   female  44.000  0         no      southwest  13063.88300    1
894    62   male    32.110  0         no      northeast  13555.00490    1
                                                                      ..
445    45   female  33.100  0         no      southwest  7345.08400     1
444    56   male    26.695  1         yes     northwest  26109.32905    1
443    59   female  36.520  1         no      southeast  28287.89766    1
442    18   male    43.010  0         no      southeast  1149.39590     1
1337   61   female  29.070  0         yes     northwest  29141.36030    1
Length: 1338, dtype: int64
```

[17] df['children'].value_counts()

```
0    574
1    324
2    240
3    157
4     25
5     18
Name: children, dtype: int64
```

[18] df_smoker = df[df['smoker'] == 'yes']
df_smoker.head()

	index	age	sex	bmi	children	smoker	region	charges
0	0	19	female	27.90	0	yes	southwest	16884.9240
11	11	62	female	26.29	0	yes	southeast	27808.7251
14	14	27	male	42.13	0	yes	southeast	39611.7577
19	19	30	male	35.30	0	yes	southwest	36837.4670
23	23	34	female	31.92	1	yes	northeast	37701.8768

[19] df_smoker.describe()

	index	age	bmi	children	charges
count	274.000000	274.000000	274.000000	274.000000	274.000000
mean	672.470803	38.514599	30.708449	1.113139	32050.231832
std	406.050495	13.923186	6.318644	1.157066	11541.547176
min	0.000000	18.000000	17.195000	0.000000	12829.455100
25%	293.000000	27.000000	26.083750	0.000000	20826.244213
50%	685.500000	38.000000	30.447500	1.000000	34456.348450
75%	1032.500000	49.000000	35.200000	2.000000	41019.207275
max	1337.000000	64.000000	52.580000	5.000000	63770.428010

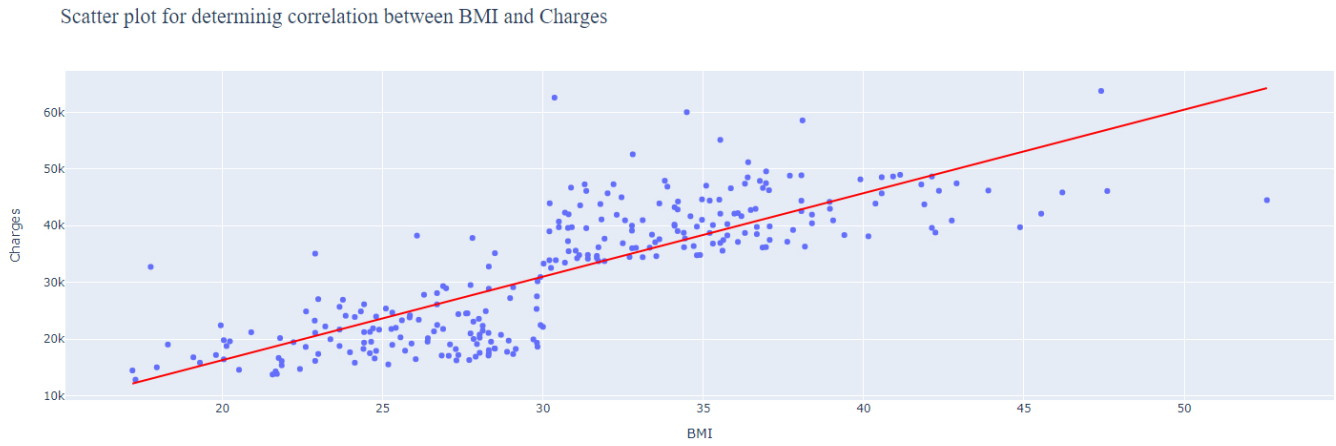
Generate Box Plot

[20] import plotly.express as pl
plt = pl.box(df_smoker, y='charges', title="Box plot of Charges for Smokers", notched = True, points = 'all')
plt.show()

The box plot of charges for smokers from the insurance dataset provides a visual representation of the distribution of charges for individuals who smoke. The box plot shows that the median charge for smokers is 34.46k, and the interquartile range (IQR) for smokers is 20.26k. Additionally, the upper whisker of the box plot extends further for smokers to 64k. Overall, the box plot suggests that smoking may be a significant factor in determining healthcare charges and may warrant further investigation as a potential risk factor

Generate Scatter Plot

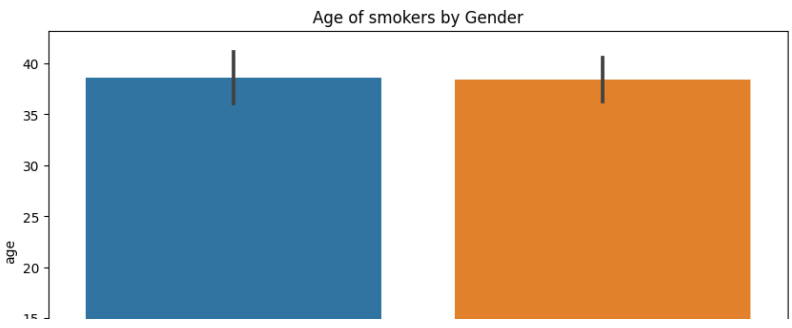
```
[21] plt = pl.scatter(df_smoker, x='bmi', y='charges',
                      labels={'bmi': 'BMI', 'charges': 'Charges'}, title='Scatter plot for determinig correlation between BMI and Charges',
                      trendline='ols', trendline_color_override='red')
     plt.update_layout(title_font_family="Times New Roman", title_font_size=22)
     plt.show()
```

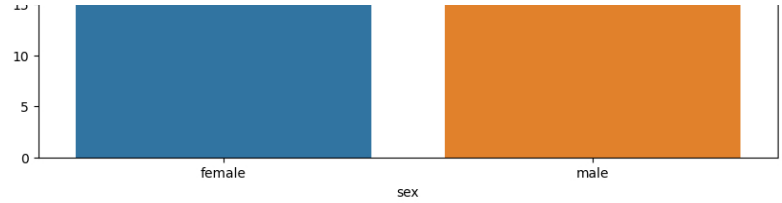


The above scatter plot provides a visual representation of the linear relationship between BMI and healthcare charges. The plot shows a positive correlation between BMI and charges, indicating that as BMI increases, healthcare charges tend to increase as well. The slope of the regression line is positive, suggesting that for every unit increase in BMI, there is a corresponding increase in healthcare charges. The plot also shows that there is some variability in the data, with some data points deviating from the regression line, indicating that other factors besides BMI may also contribute to healthcare charges. Overall, the graph supports the conclusion that BMI is a significant predictor of healthcare charges and may be useful in predicting charges for individuals based on their BMI. However, additional analysis is needed to identify other factors that may contribute to healthcare charges and to develop a more comprehensive model for predicting charges.

Generate Bar Plot

```
[22] import matplotlib.pyplot as mpl
     import seaborn as sns
     mpl.figure(figsize=(10,6))
     sns.barplot(x='sex',y='age',data=df_smoker)
     mpl.xticks()
     mpl.title("Age of smokers by Gender")
     mpl.show()
```





The above bar plot provides a visual representation of the number of smokers by gender and age group. The plot shows the highest number smokers are in the age group of 35-40 years for both the gender. Overall, the bar plot suggests that gender and age are important factors to consider when studying smoking behavior. However, additional research is needed to better understand the complex factors that contribute to smoking behavior.

Ridge Regression

```
[23] #import pandas as pd
     import numpy as np
     import sklearn
     import warnings
     warnings.filterwarnings("ignore")
     from sklearn.model_selection import train_test_split
     from sklearn.linear_model import LinearRegression
```

Splitting the data as training and testing dataset

```
[24] x= df['bmi']
     y=df['charges']
     x = np.array([x]).reshape((-1, 1))
     y = np.array([y]).reshape((-1, 1))
     x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25)
```

Creating Linear Regression model

```
[25] mod = LinearRegression().fit(x_train,y_train)
```

```
[26] y_predict = mod.predict(x_test)
     y_predict
```

```
array([[11608.72224325],
       [13530.52764048],
       [17026.22538888],
       [16018.93428412],
       [11535.82617646],
       [17403.95955316],
       [12082.5466774 ],
       [13175.98767926],
       [12871.14958177],
       [13563.66221629],
       [12447.02701135],
       [16893.68708562],
       [14524.5649149 ],
       [13532.18436927],
       [15483.81088472],
       [11171.34584251],
       [12735.29782094],
       [12039.47172884],
       [13066.64357908],
       [14004.35207462],
       [16091.83035091],
       [ 9686.91684603],
       [11202.82368953],
       [11443.04936418],
       [13563.66221629],
       [14670.35704849],
       [13895.00797443],
       [ 9720.05142184],
       [12556.37111154],
       [14524.5649149 ],
       [10415.87751394],
       [13613.36408001],
       [10916.20960874],
       [12808.19388773],
       [15982.48625073],
       [14358.89203583],
       [11645.17027665],
       [12084.20340619],
       [14287.65269783],
       [15472.21378319],
       [11234.30153655],
       [15011.64317937],
       [10195.53258478],
       [12702.16324512],
       [11706.4692419 ],
       [13686.2601468 ],
       [11840.66427395],
       [15849.94794747],
       [11769.42493595],
       [12867.83612419],
       [11832.38063   ],
       [14728.34255616],
       [11718.06634344],
       [14723.37236979],
       [12147.15910023],
       [14160.08458095],
       [15816.81337166],
       [10227.0104318 ]])
```

```
[27] from sklearn.metrics import r2_score, mean_squared_error
     print('R2 score',r2_score(y_test, y_predict))
     display('Mean squared error', np.sqrt(mean_squared_error(y_test, y_predict)))
```

```
R2 score 0.05236080441102675
'Mean squared error'
12641.920803742189
```

Creating Ridge Regression model

```
[28] from sklearn.linear_model import Ridge
     R = Ridge(alpha=0.1)
```

```
[29] R.fit(x_train,y_train)
```

Ridge(alpha=0.1)

```
[30] y_predict_rr = R.predict(x_test)
```

```
[31] ridge_r_square = r2_score(y_test,y_predict_rr)
     display ("R-square value is", ridge_r_square)
     ridge_msv = np.sqrt(mean_squared_error(y_test,y_predict_rr))
     display("Mean Square value is", ridge_msv)
```

```
'R-square value is'
0.05236070833799644
'Mean Square value is'
12641.921444570267
```

Grid Search

```
[32] from sklearn.model_selection import GridSearchCV
     parameters= [{'alpha': [0.001,0.1,1, 10, 100, 1000, 10000, 100000, 100000]}]
     parameters
```

```
[{'alpha': [0.001, 0.1, 1, 10, 100, 1000, 10000, 100000, 100000]}]
```

```
[33] RR=Ridge()
     RR
```

Ridge()

```
[34] Grid = GridSearchCV(RR, parameters,cv=4)
```

```
[35] Grid.fit(x,y)
```

GridSearchCV
estimator: Ridge
Ridge

```
[36] BestRR = Grid.best_estimator_
     BestRR
```

Ridge(alpha=1000)

```
[37] BestRR.fit(x_train, y_train)
```

Ridge(alpha=1000)

```
[38] y_predgrid = BestRR.predict(x_test)
     y_predgrid
```

```
[11644.31444854],
[13517.06849388],
[16923.54352461],
[15941.96209395],
[11573.27895027],
[17291.63656111],
[12106.04518731],
[13171.57766138],
[12874.52012315],
[13549.35735673],
[12461.22267866],
[16794.38807321],
[14485.73437939],
[13518.68293702],
[15420.49695892],
[11218.10145892],
[12742.13578546],
[12064.0696656 ],
[13065.02441397],
[13978.79923264],
[16012.99759222],
[ 9771.56040321],
[11248.77587863],
[11482.87013429],
[13549.35735673],
[14627.80537594],
[13872.24598523],
[ 9803.84926606],
[12567.77592607],
[14485.73437939],
[10481.91538592],
[13597.790651  ],
[10969.47721497],
[12813.17128373],
```

```
[15966.44434482],
[14324.29006514],
[11679.83219768],
[12107.65963045],
[14254.86901001],
[15409.19585692],
[11279.45029833],
[14960.3806633 ],
[10267.19444797],
[12709.84692261],
[11739.56659395],
[13668.82614927],
[11870.3364885 ],
[15777.28889342],
[11800.91543337],
[12871.29123687],
[11862.26427279],
[14684.31088592],
[11750.86769595],
[14679.4675565 ],
[12169.00846987],
[14130.55688804],
[15745.00003056],
[10297.86886768]])
```

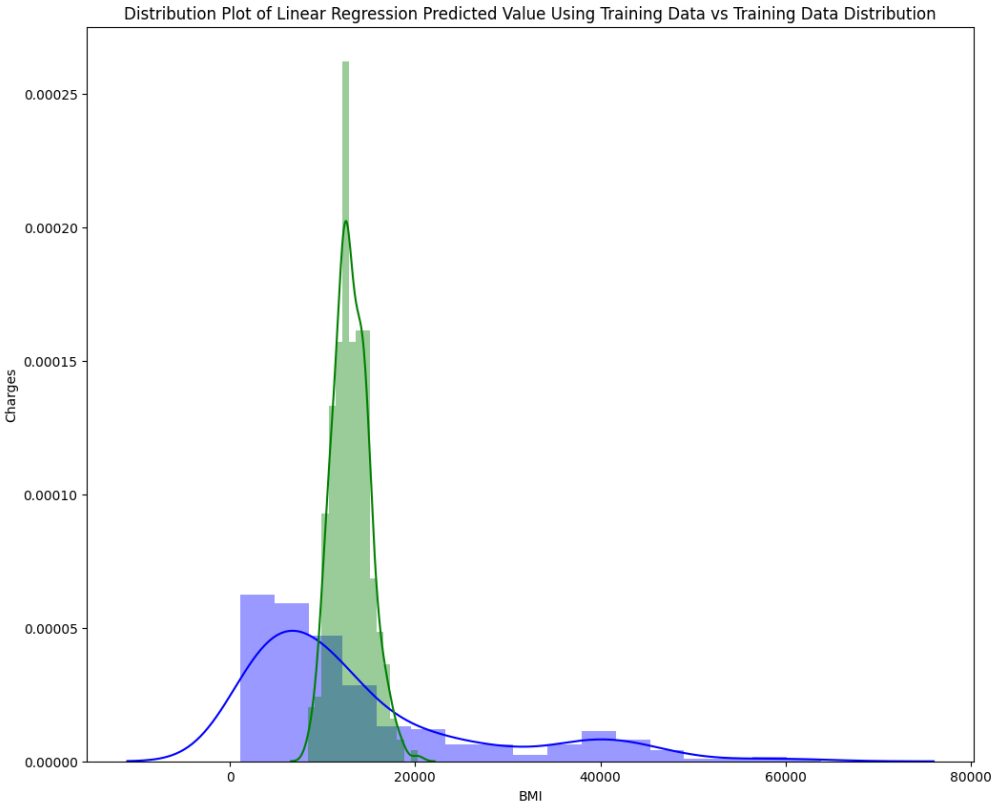
[39] `BestRR.score(x,y)`

```
0.03763660151284354
```

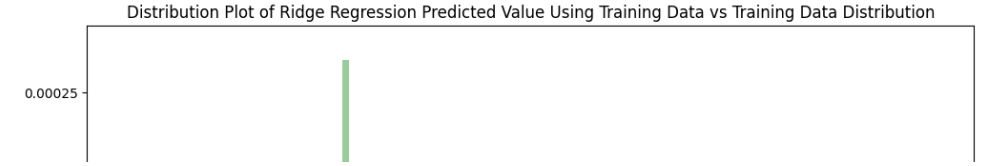
Generate Distribution Plots

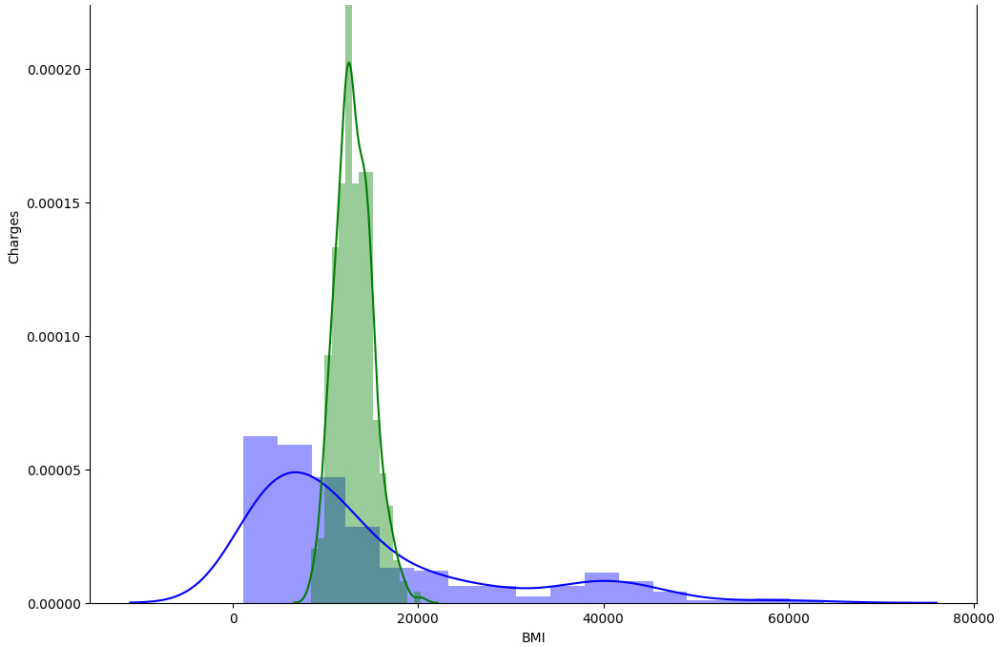
[40]
```
%matplotlib inline
def DistributionPlot(BlueFunction, GreenFunction, RedName, BlueName, Title):
    width = 12
    height = 10
    mpl.figure(figsize=(width, height))

    ax1 = sns.distplot(BlueFunction, color="blue", label=RedName)
    ax2 = sns.distplot(GreenFunction, color="green", label=BlueName, ax=ax1)

    mpl.title(Title)
    mpl.xlabel('BMI')
    mpl.ylabel('Charges')

    mpl.show()
    mpl.close()
```

[41]
```
Title = 'Distribution Plot of Linear Regression Predicted Value Using Training Data vs Training Data Distribution'
DistributionPlot(y_test, y_predict, "Actual Values (Train)", "Predicted Values (Train)", Title)
```

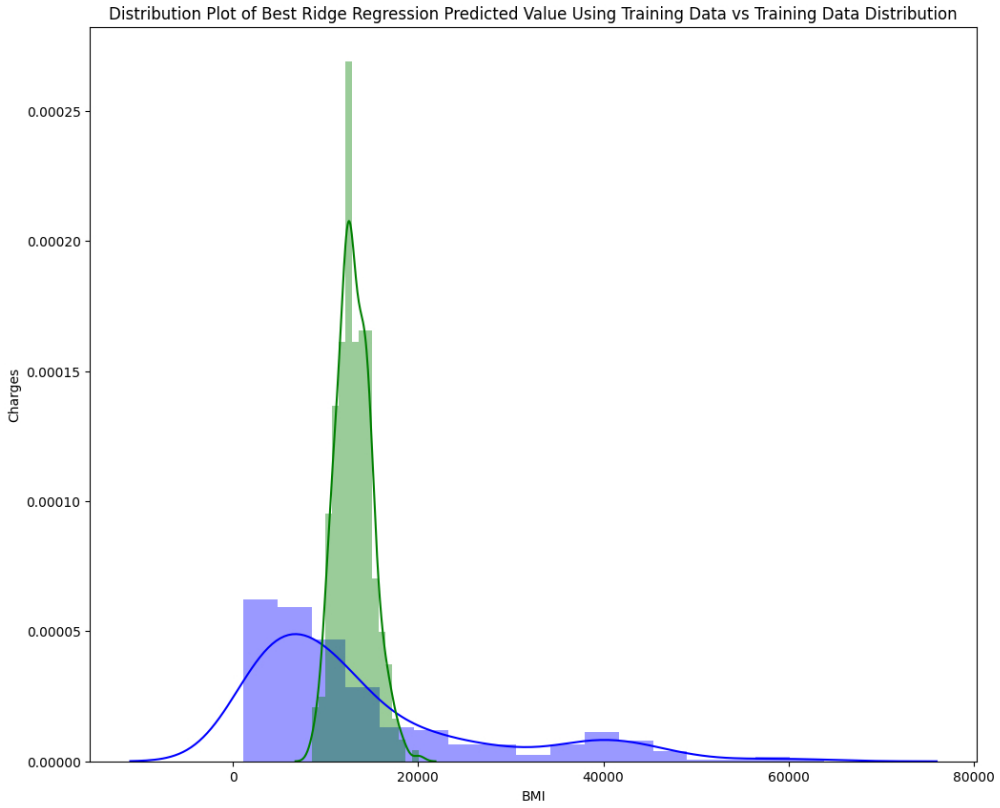


[42]
```
Title = 'Distribution Plot of Ridge Regression Predicted Value Using Training Data vs Training Data Distribution'
DistributionPlot(y_test, y_predict_rr, "Actual Values (Train)", "Predicted Values (Train)", Title)
```






```
[43] Title = 'Distribution Plot of Best Ridge Regression Predicted Value Using Training Data vs Training Data Distribution'
     DistributionPlot(y_test, y_predgrid, "Actual Values (Train)", "Predicted Values (Train)", Title)
```



Conclusion:

An R-squared value of 0.022 for both Linear Regression and Ridge Regression models indicates that only a small proportion (2.2%) of the variance in Charges can be explained by changes in BMI. This means that the linear relationship between BMI and Charges is weak.

The Grid Search model with an R-squared value of 0.039 performs slightly better than the other two models. However, the increase in R-squared value is still relatively small, indicating that the Grid Search model is also not a very good fit for the data.

In conclusion, based on the R-squared values, the models suggest that there is only a weak relationship between BMI and Charges. Other factors, such as age, gender, smoking habits may play a more significant role in determining healthcare charges. Therefore, it may be necessary to consider other independent variables to better understand the relationship between the independent and dependent variables.

✓ 0s completed at 10:07 AM