Medical Cost Prediction

> Objective: Leveraging advanced analytics to predict medical expenses based on patient information using the Kaggle Insurance dataset.

```
#importing the libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
from google.colab import drive
drive.mount ('/content/drive')
```

```
Mounted at /content/drive
```

Exploratory Data Analysis-EDA

```
df = pd.read_csv('/content/drive/MyDrive/insurance.csv')
df.head()
```

	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

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1338 non-null   int64
 1   sex       1338 non-null   object
 2   bmi       1338 non-null   float64
 3   children  1338 non-null   int64
 4   smoker    1338 non-null   object
 5   region    1338 non-null   object
 6   charges   1338 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 73.3+ KB
```

```
df.shape
```

```
(1338, 7)
```

```
print(df.columns)
```

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

```
print(df.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
```

```
print(df.dtypes)
```

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

```
#value counts for categorical variables
print(df.sex.value_counts(),'\n',df.smoker.value_counts(),'\n',df.region.value_counts())
```

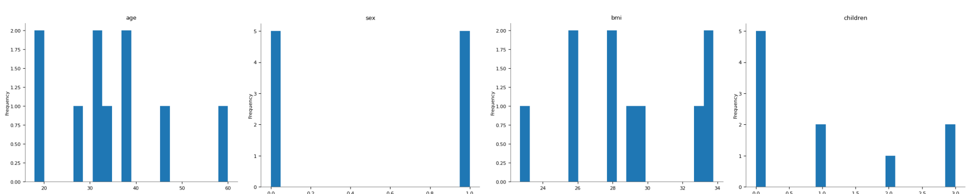
```
male      676
female    662
Name: sex, dtype: int64
 no     1064
yes     274
Name: smoker, dtype: int64
 southeast    364
southwest    325
northwest    325
northeast    324
Name: region, dtype: int64
```

```
#changing categorical variables to numerical
df['sex'] = df['sex'].map({'male':1,'female':0})
df['smoker'] = df['smoker'].map({'yes':1,'no':0})
df['region'] = df['region'].map({'southwest':0,'southeast':1,'northwest':2,'northeast':3})
```

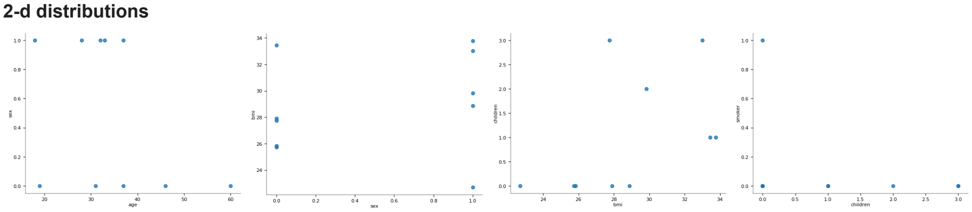
```
df.head(10)
```

	age	sex	bmi	children	smoker	region	charges
0	19	0	27.900	0	1	0	16884.92400
1	18	1	33.770	1	0	1	1725.55230
2	28	1	33.000	3	0	1	4449.46200
3	33	1	22.705	0	0	2	21984.47061
4	32	1	28.880	0	0	2	3866.85520
5	31	0	25.740	0	0	1	3756.62160
6	46	0	33.440	1	0	1	8240.58960
7	37	0	27.740	3	0	2	7281.50560
8	37	1	29.830	2	0	3	6406.41070
9	60	0	25.840	0	0	2	28923.13692

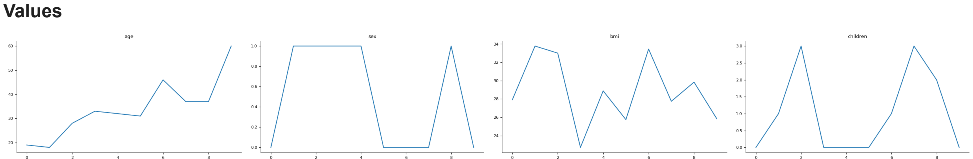
Distributions



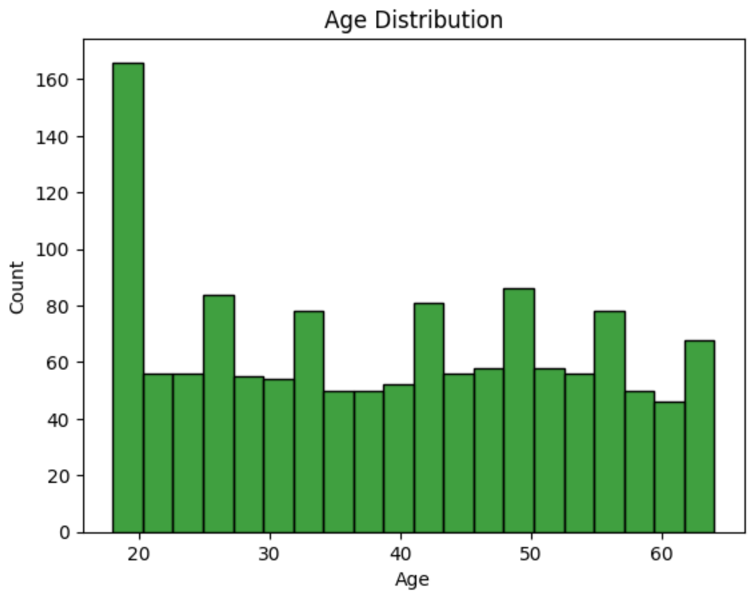
2-d distributions



Values



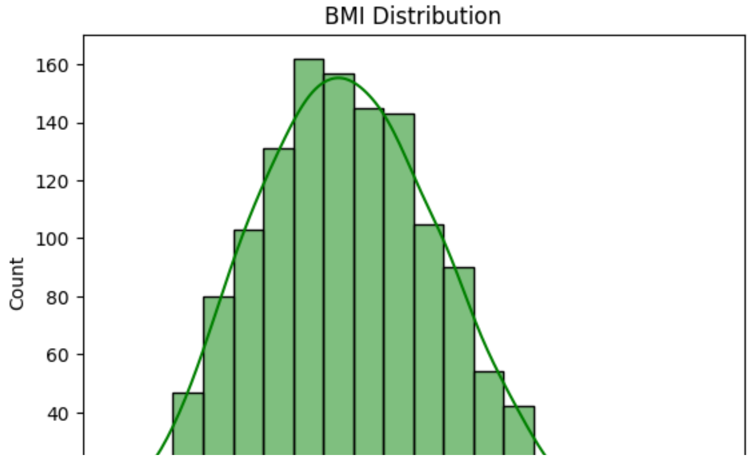
```
#age distribution
sns.histplot(df.age,bins=20, kde=False,color='green')
plt.title('Age Distribution')
plt.xlabel('Age')
plt.ylabel('Count')
plt.show()
```



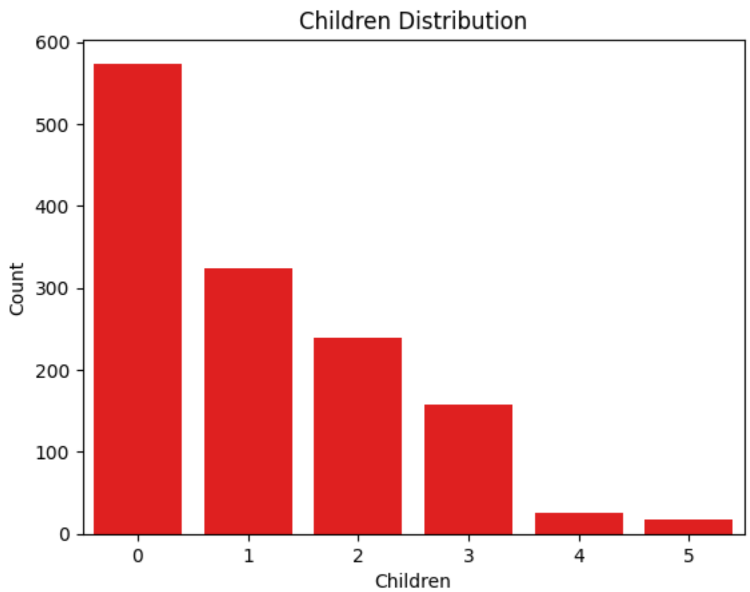

```
#gender plot
sns.countplot(x = 'sex', data = df)
plt.title('Gender Distribution')
```

```
Text(0.5, 1.0, 'Gender Distribution')
```

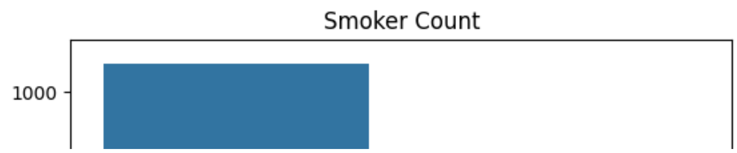

```
#bmi distribution
sns.histplot(df.bmi,bins=20, kde=True,color='green')
plt.title('BMI Distribution')
plt.xlabel('BMI')
plt.ylabel('Count')
plt.show()
```



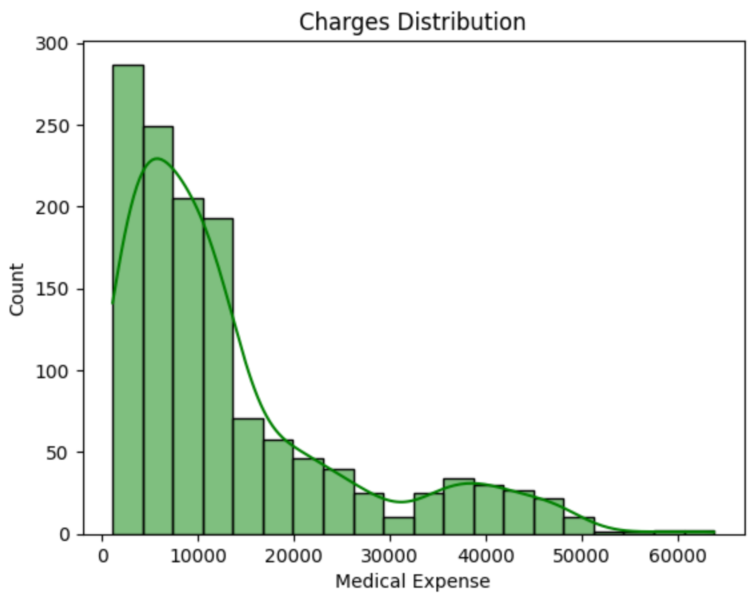

```
#child count distribution
sns.countplot(x = 'children', data = df, color="red")
plt.title('Children Distribution')
plt.xlabel('Children')
plt.ylabel('Count')
plt.show()
```




```
#count of smokers
sns.countplot(x = 'smoker', data = df)
plt.title('Smoker Count')
plt.xlabel('Smoker')
plt.ylabel('Count')
plt.show()
```




```
#charges distribution
sns.histplot(df.charges,bins=20, kde=True,color='green')
plt.title('Charges Distribution')
plt.xlabel('Medical Expense')
plt.ylabel('Count')
plt.show()
```

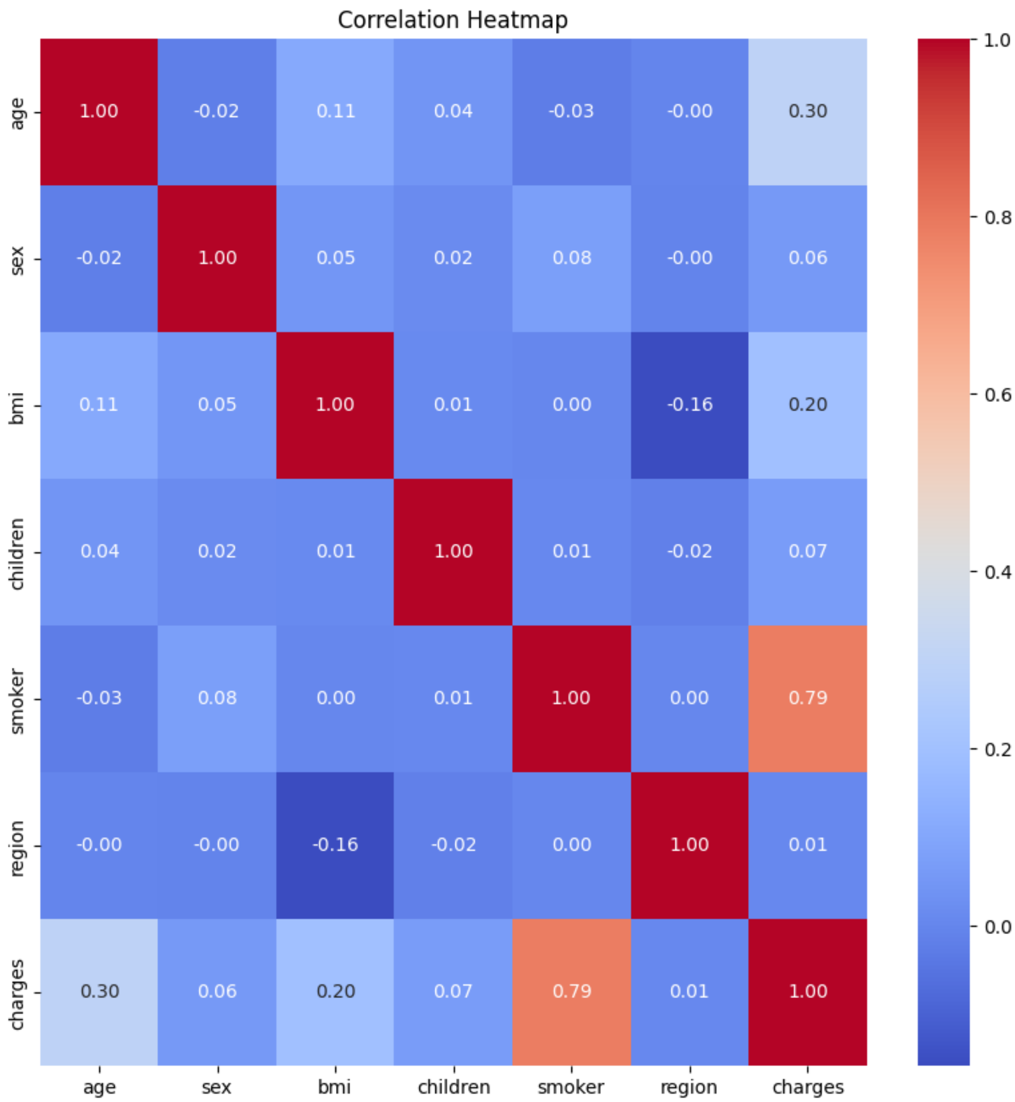


Coorelation

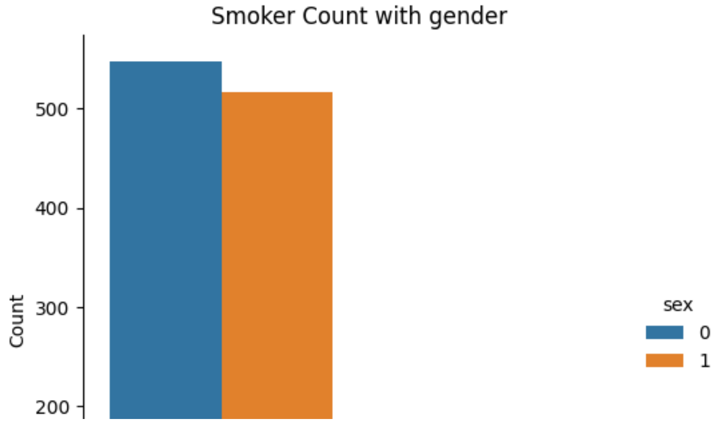
```
#coorelation matrix
df.corr()
```

	age	sex	bmi	children	smoker	region	charges
age	1.000000	-0.020856	0.109272	0.042469	-0.025019	-0.002127	0.299008
sex	-0.020856	1.000000	0.046371	0.017163	0.076185	-0.004588	0.057292
bmi	0.109272	0.046371	1.000000	0.012759	0.003750	-0.157566	0.198341
children	0.042469	0.017163	0.012759	1.000000	0.007673	-0.016569	0.067998

```
#plotting the coorelation heatmap
plt.figure(figsize=(10,10))
sns.heatmap(df.corr(),annot=True,cmap='coolwarm',fmt=".2f")
plt.title("Correlation Heatmap")
plt.show()
```

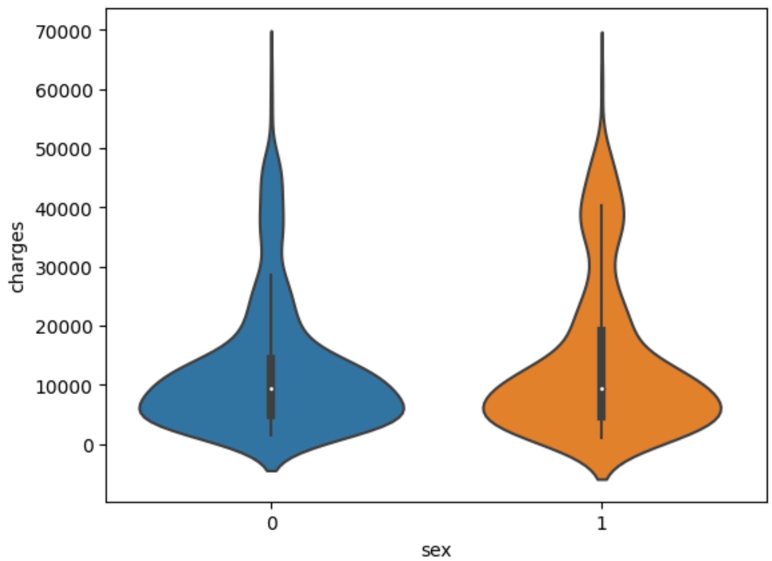



```
sns.catplot(x="smoker", kind="count",hue = 'sex', data=df)
plt.title('Smoker Count with gender')
plt.xlabel('Smoker')
plt.ylabel('Count')
plt.show()
```



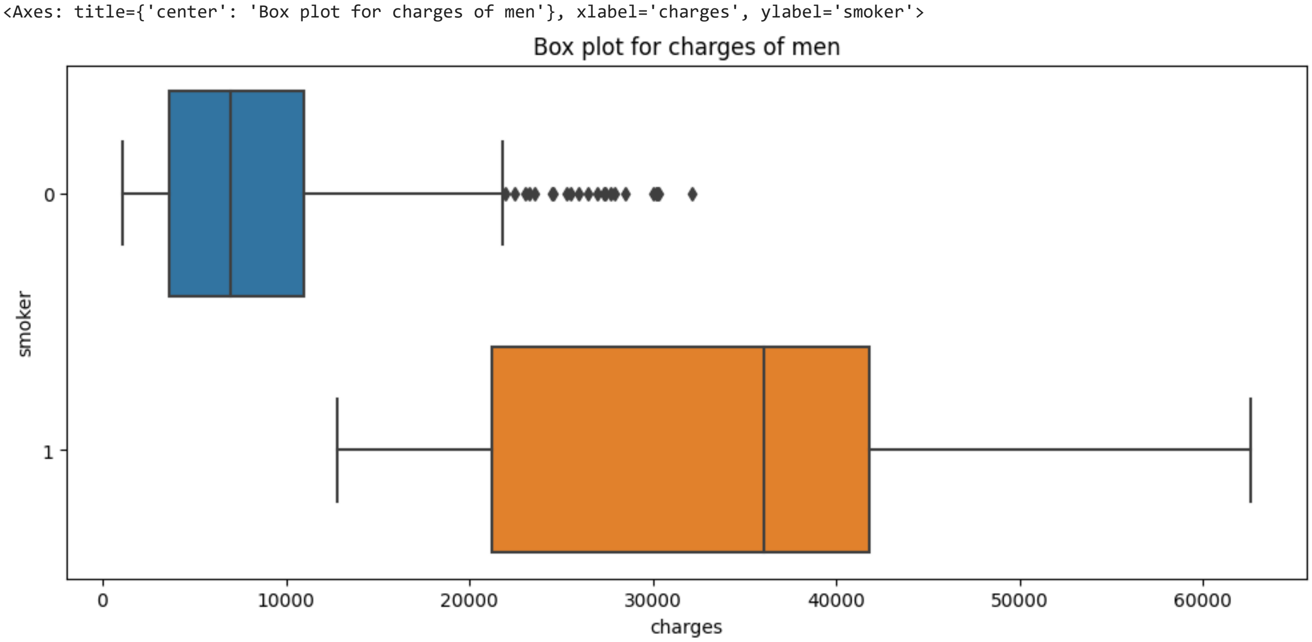

```
sns.violinplot(x = 'sex', y = 'charges', data = df)
```

<Axes: xlabel='sex', ylabel='charges'>



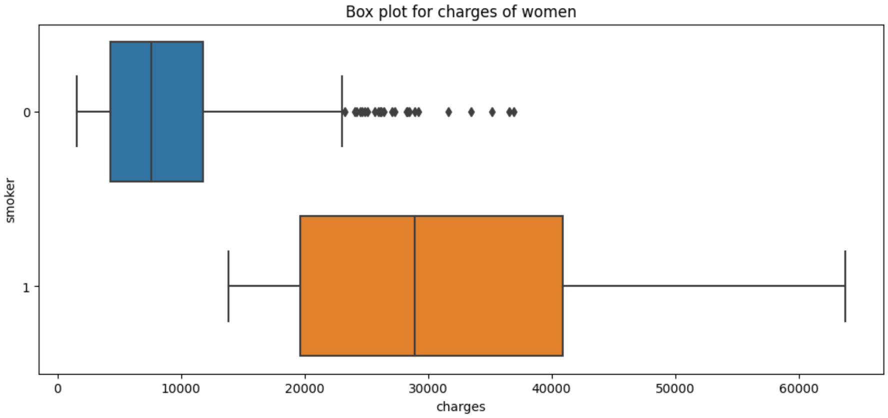

```
plt.figure(figsize=(12,5))
plt.title("Box plot for charges of men")
sns.boxplot(y="smoker", x="charges", data =  df[(df.sex == 1)] , orient="h")
```

<Axes: title={'center': 'Box plot for charges of men'}, xlabel='charges', ylabel='smoker'>



```
plt.figure(figsize=(12,5))
plt.title("Box plot for charges of women")
sns.boxplot(y="smoker", x="charges", data =  df[(df.sex == 0)] , orient="h")
```

```
<Axes: title={'center': 'Box plot for charges of women'}, xlabel='charges',
ylabel='smoker'>
```



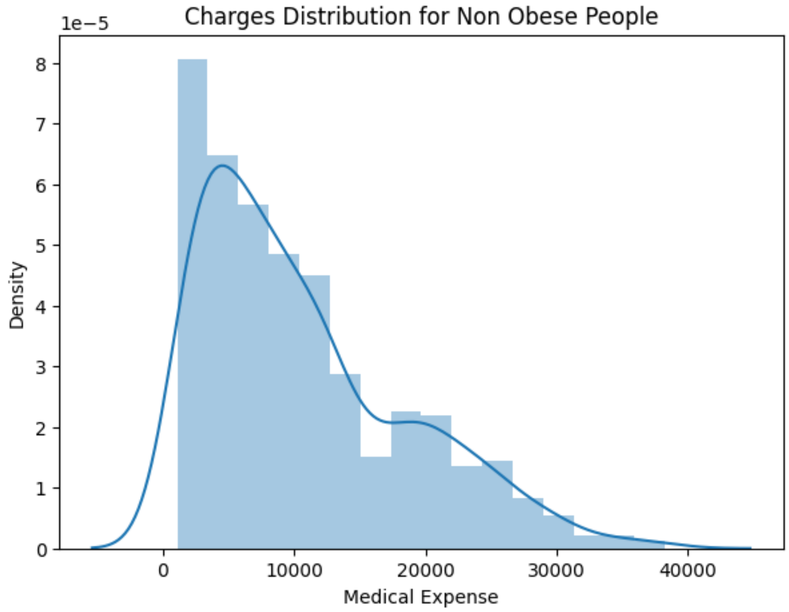

```
#bmi charges distribution for obese people
plt.figure(figsize=(7,5))
sns.distplot(df[(df.bmi >= 30)]['charges'])
plt.title('Charges Distribution for Obese People')
plt.xlabel('Medical Expense')
plt.show()
```

```
<ipython-input-29-1572e034011d>:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df[(df.bmi >= 30)]['charges'])
```


```
plt.figure(figsize=(7,5))
sns.distplot(df[(df.bmi < 30)]['charges'])
```

```
plt.title('Charges Distribution for Non Obese People')
plt.xlabel('Medical Expense')
plt.show()
```

```
<ipython-input-30-1ad75a47c6a0>:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df[(df.bmi < 30)]['charges'])
```



Model Training

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(df.drop('charges',axis=1), df['charges'], test_size=0.2, random_state=0)
```

```
#Linear Regression
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr
```

```
▾ LinearRegression
LinearRegression()
```

```
#model training
lr.fit(x_train,y_train)
#model accuracy
lr.score(x_train,y_train)
```

```
0.7368306228430945
```

```
#model prediction
y_pred = lr.predict(x_test)
```

Polynomial Regression

```
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree=2)
poly_reg
```

```
▾ PolynomialFeatures
PolynomialFeatures()
```

```
#transforming the features to higher degree
x_train_poly = poly_reg.fit_transform(x_train)
#splitting the data
x_train, x_test, y_train, y_test = train_test_split(x_train_poly, y_train, test_size=0.2, random_state=0)
```

```
plr = LinearRegression()
#model training
plr.fit(x_train,y_train)
#model accuracy
plr.score(x_train,y_train)
```

```
0.836373486593943
```

```
#model prediction
y_pred = plr.predict(x_test)
```

Decision Tree Regressor

```
#decision tree regressor
from sklearn.tree import DecisionTreeRegressor
dtree = DecisionTreeRegressor()
dtree
```

```
▾ DecisionTreeRegressor
DecisionTreeRegressor()
```

```
#model training
dtree.fit(x_train,y_train)
#model accuracy
dtree.score(x_train,y_train)
```

```
0.9993688476658964
```

```
#model prediction
dtree_pred = dtree.predict(x_test)
```

Random Forest Regressor

```
#random forest regressor
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(n_estimators=100)
rf
```

```
▾ RandomForestRegressor
RandomForestRegressor()
```

```
#model training
rf.fit(x_train,y_train)
#model accuracy
rf.score(x_train,y_train)
```

```
0.9753148248674263
```

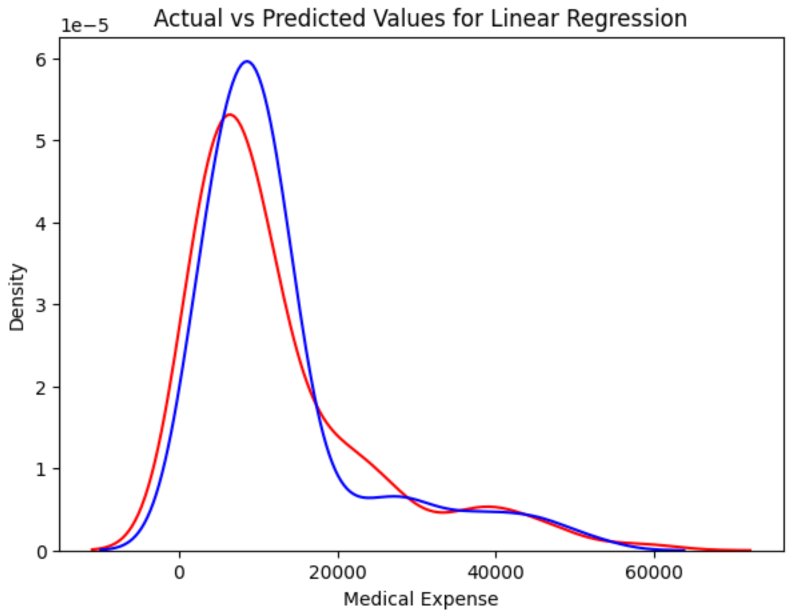
```
#model prediction
rf_pred = rf.predict(x_test)
```

Model Evaluation

```
from sklearn.metrics import mean_squared_error,mean_absolute_error,r2_score
```

```
#distribution of actual and predicted values
plt.figure(figsize=(7,5))
ax1 = sns.distplot(y_test,hist=False,color='r',label='Actual Value')
sns.distplot(y_pred,hist=False,color='b',label='Predicted Value',ax=ax1)
plt.title('Actual vs Predicted Values for Linear Regression')
plt.xlabel('Medical Expense')
plt.show()
```

```
<ipython-input-46-2d0e63236188>:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  ax1 = sns.distplot(y_test,hist=False,color='r',label='Actual Value')
<ipython-input-46-2d0e63236188>:4: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(y_pred,hist=False,color='b',label='Predicted Value',ax=ax1)
```

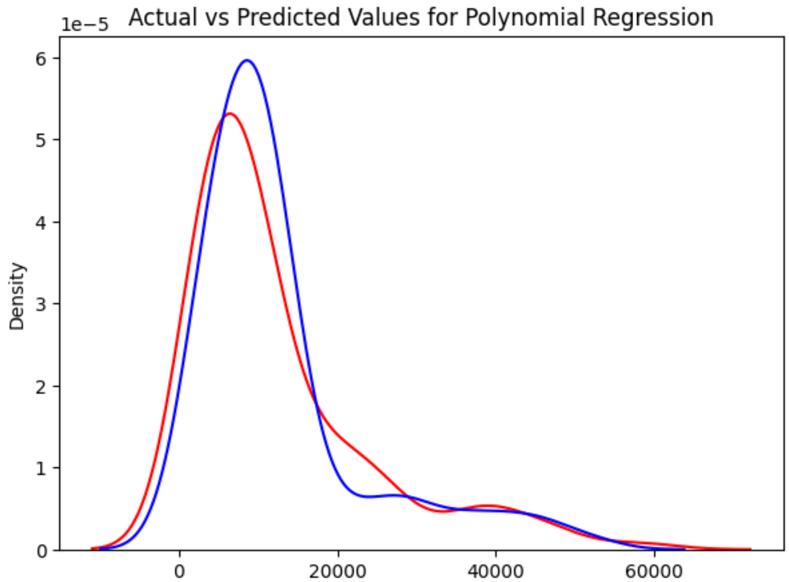



```
print('MAE:', mean_absolute_error(y_test, y_pred))
print('MSE:', mean_squared_error(y_test, y_pred))
print('RMSE:', np.sqrt(mean_squared_error(y_test, y_pred)))
print('R2 Score:', r2_score(y_test, y_pred))
```

```
MAE: 3016.8193118925233
MSE: 24705741.734187007
RMSE: 4970.48707212754
R2 Score: 0.8207480676082507
```

```
#acutal vs predicted values for polynomial regression
plt.figure(figsize=(7,5))
ax1 = sns.distplot(y_test,hist=False,color='r',label='Actual Value')
sns.distplot(y_pred,hist=False,color='b',label='Predicted Value',ax=ax1)
plt.title('Actual vs Predicted Values for Polynomial Regression')
plt.xlabel('Medical Expense')
plt.show()
```

```
<ipython-input-48-7a574536b1bb>:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  ax1 = sns.distplot(y_test,hist=False,color='r',label='Actual Value')
<ipython-input-48-7a574536b1bb>:4: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(y_pred,hist=False,color='b',label='Predicted Value',ax=ax1)
```

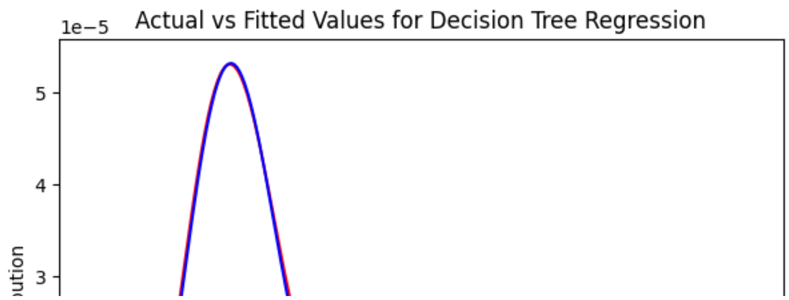



```
print('MAE:', mean_absolute_error(y_test, y_pred))
print('MSE:', mean_squared_error(y_test, y_pred))
print('RMSE:', np.sqrt(mean_squared_error(y_test, y_pred)))
print('R2 Score:', r2_score(y_test, y_pred))
```

```
MAE: 3016.8193118925233
MSE: 24705741.734187007
RMSE: 4970.48707212754
R2 Score: 0.8207480676082507
```

```
#distribution plot of actual and predicted values
plt.figure(figsize=(7,5))
ax = sns.distplot(y_test, hist=False, color="r", label="Actual Value")
sns.distplot(dtree_pred, hist=False, color="b", label="Fitted Values" , ax=ax)
plt.title('Actual vs Fitted Values for Decision Tree Regression')
plt.xlabel('Medical Expense')
plt.ylabel('Distribution')
plt.show()
```

```
<ipython-input-50-46f60f40ec0e>:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  ax = sns.distplot(y_test, hist=False, color="r", label="Actual Value")
<ipython-input-50-46f60f40ec0e>:4: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(dtree_pred, hist=False, color="b", label="Fitted Values" , ax=ax)
```




```
print('MAE:', mean_absolute_error(y_test, rf_pred))
print('MSE:', mean_squared_error(y_test, rf_pred))
print('RMSE:', np.sqrt(mean_squared_error(y_test, rf_pred)))
print('Accuracy:', rf.score(x_test,y_test))
```

```
MAE: 2823.0877917939247
MSE: 26490443.801917486
RMSE: 5146.886806790828
Accuracy: 0.8077992034200706
```