Georgia State University

J. Mack Robinson College of Business

CIS 8005 Group Project Team 1

Medical Expenses Prediction

Gopi Sandeep Guntuku: gguntuku1@student.gsu.edu

Sukumar Reddy Kottala: skottala1@student.gsu.edu

Project Summary

Project description:

Everyone's life revolves around their health. Good health is essential to all aspects of our lives. Health refers to a person's ability to cope up with the environment on a physical, emotional, mental, and social level.

Because of the quick speed of our lives, we are adopting many habits that are harming our health. One spends a lot of money to be healthy by participating in physical activities or having frequent health check-ups to avoid being unfit and get rid of health disorders. When we become ill, we tend to spend a lot of money, resulting in a lot of medical expenses.

So, an application can be made which can make people understand the factors which are making them unfit, and creating a lot of medical expenses, and it could identify and estimate medical expense if someone has such factors.

Project description:

This project aims at building Machine Learning models which can predict a patient's medical expenses based on specific features and identifying the factors affecting the medical expenses of the subjects based on the model output.

Factors affecting the medical expenses of the patients :

- **Age**
- **Gender**
- **Body Mass Index**
- **Region**
- **Smoking Behavior**

.

Business Implications of the Project:

- Health is the center of everyone's life.
- Every part of our life relies on good health.
- Health is the extent of an individual's continuing physical, emotional, mental, and social ability to cope with the environment.

Data Source:

- The dataset has been sourced from **kaggle.com** and on **GitHub**.
- This data file includes all needed information to find out more about age, gender, smoking behavior and necessary metrics to make predictions and draw conclusions.
- **Link to the dataset: https://github.com/stedy/Machine-Learning-with-R-datasets/blob/master/insurance.csv**

The dataset contains information on 1,338 patients.

It includes the following features:

- Age: Patient's age
- Sex: Patient's sex
- BMI: Patient's Body Mass Index
- Children: How many children the patient has
- Smoker: Whether the patient is a smoker or not
- Region: Which region the patient is from (Northeast, Southeast, Southwest, Northwest)
- Expenses: Individual medical costs billed by health insurance

Target Variable:

The Expenses column is the target column, and the rest others are independent columns. Independent columns are those which will predict the outcome.

Independent Variables:

The first column is Age. Age is an important factor for predicting medical expenses because young people are generally more healthy than old ones and the medical expenses for Young People will be quite less as compared to old people.

The Next column is sex, which has two Categories in this column: Male and Female. The sex of the person can also play a vital role in predicting the medical expenses of a subject.

After that, you have the 'bmi' column, then **BMI is Body Mass Index.** For most adults, an ideal BMI is in the 18.5 to 24.9 range. For children and young people aged 2 to 18, the BMI calculation considers age and gender as well as height and weight. If your BMI is less than 18.5, you are considered underweight. People with very low or very high 'bmi' are more likely to require medical assistance, resulting in higher costs.

The fourth column is the ‘children’ column, which contains information on how many children your patients have. Persons who have children are under more pressure because of their children’s education, and other needs than people who do not have children.

The fifth is the ‘smoker’ column. The Smoking factor is also considered to be one of the Most Important factors as the people who smoke are always at risk when their age reaches 50 to 60.

Next is the ‘region’ column. Some Regions are Hygienic, Clean, Neat, and Prosperous, But some Regions are not, and this information affects health which is related to medical expenses.

Steps:

- Data Pre-Processing
- Exploratory Data Analysis
    - Data Visualization
    - Univariate Analysis
    - Data Processing
- Data Modelling
- Evaluation
- Conclusion
- Recommendations
- Future Work

A snippet of how the data looks like:

	age	sex	bmi	children	smoker	region	expenses
0	19	female	27.9	0	yes	southwest	16884.92
1	18	male	33.8	1	no	southeast	1725.55
2	28	male	33.0	3	no	southeast	4449.46
3	33	male	22.7	0	no	northwest	21984.47
4	32	male	28.9	0	no	northwest	3866.86
...	...	...	...	...	...	...	...
1333	50	male	31.0	3	no	northwest	10600.55
1334	18	female	31.9	0	no	northeast	2205.98
1335	18	female	36.9	0	no	southeast	1629.83
1336	21	female	25.8	0	no	southwest	2007.95
1337	61	female	29.1	0	yes	northwest	29141.36

1337 rows × 7 columns

We'll be using the dataset to model and predict the below:

- *Performing EDA on the dataset [Data Pre-Processing and Data Visualization]:*
    - Predict the price of the listings by employing **Linear Regression, K-Nearest Neighbors Regression, Random Forest Regression** after proper data processing, label encoding, normalization & feature engineering.

Platform used: Google Collab

Data Preparation

Data Pre-Processing:

- Importing the required libraries for mathematical operations, data frame manipulations and data visualizations. Read the dataset.

```
# Import the required libraries

# For mathemaical operations
import numpy as np
# For dataframe manipulations
import pandas as pd

# For data visualizations
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px

# Setting parameters for visualization
plt.rcParams['figure.figsize'] = (20, 10)
plt.style.use('fivethirtyeight')
```

```
[ ] # Read the data set

data = pd.read_csv('med-expense.csv')

# Total number of rows and columns

data.shape
```

```
(1338, 7)
```

So here we have total 1338 Rows and 7 Columns in the given dataset

- Display all the columns with corresponding count of non-null observation and along with their data types

```
[ ] # Display all the columns with corresponding count of non null observation and along with their data types

data.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   expenses  1338 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 73.3+ KB
```

We observed that "age", "children", "bmi" and "expenses" are numbers, whereas "gender", "smoker" and "region" are strings (possibly categories).

None of the columns contain any missing values, which saves us a fair bit of work.

- Statistical Measurement of Variables

```
[ ]  # Statistical measurement of variables

     data.describe()
```

	age	bmi	children	expenses
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.665471	1.094918	13270.422414
std	14.049960	6.098382	1.205493	12110.011240
min	18.000000	16.000000	0.000000	1121.870000
25%	27.000000	26.300000	0.000000	4740.287500
50%	39.000000	30.400000	1.000000	9382.030000
75%	51.000000	34.700000	2.000000	16639.915000
max	64.000000	53.100000	5.000000	63770.430000

From the above dataset we can see that Age, BMI, Children, Expenses are only Numeric Values and get their corresponding statistical measurement.

Here from the above information, we can conclude that Age variable is symmetrically distributed as Mean = Median.

BMI is also symmetrically distributed.

But Expenses variable is positively skewed as Mean > Median.

- Removing the Null values from the dataset using isnull().sum() & use dropna() if there are any null values.

```
# Checking for null values

data.isnull().sum()
```

```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
expenses    0
dtype: int64
```

So, here there are no null values with respect to each Column of the dataset. Hence data cleaning is not required.

- Check for number of unique values.

```
[ ]  # Checking for number of unique values

     data.nunique()
```

```
age           47
sex            2
bmi          275
children       6
smoker         2
region         4
expenses    1337
dtype: int64
```

- Check for duplicates using duplicated() and drop the second duplicated row and check the count after dropping duplicate.

As the expenses column consists of total of 1338 values, these seems to be one duplicate value.

```
# Lets remove the duplicate value

duplicate_rows_data = data[data.duplicated()]
duplicate_rows_data
```

	age	sex	bmi	children	smoker	region	expenses
581	19	male	30.6	0	no	northwest	1639.56

```
[ ]  # Identifying the duplicated row based on the expense

     data[data["expenses"] == 1639.56]
```

	age	sex	bmi	children	smoker	region	expenses
195	19	male	30.6	0	no	northwest	1639.56
581	19	male	30.6	0	no	northwest	1639.56

```
[ ]  # Dropping the second duplicate row

     data.drop_duplicates(keep = 'first', inplace = True)
     data[data.duplicated()]
```

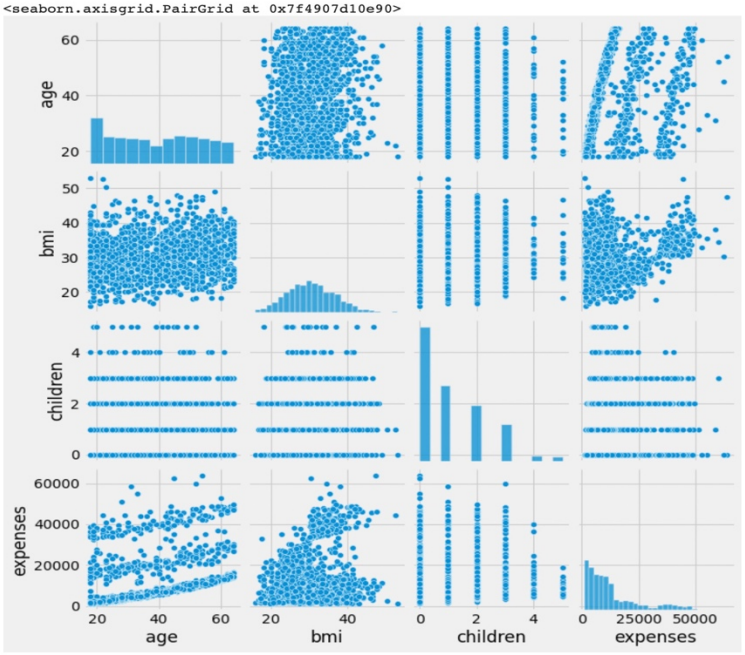
	age	sex	bmi	children	smoker	region	expenses

Exploratory Data Analysis:

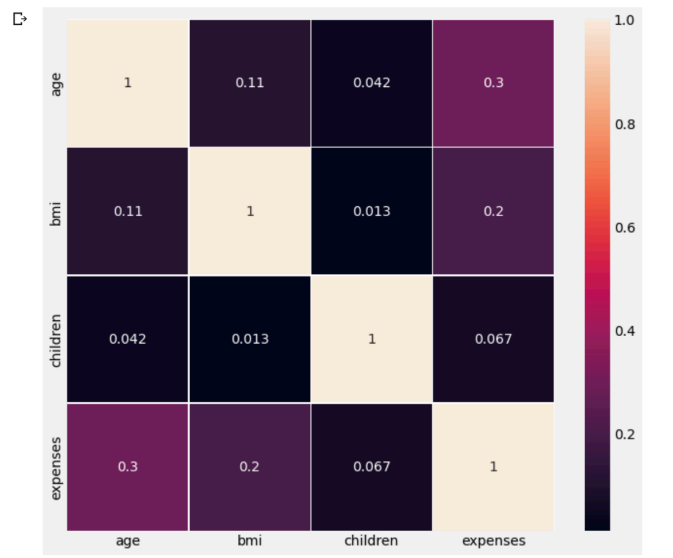
Data Visualization:

```
# Visualizing using Pair Plot

sns.pairplot(data)
```

```
<seaborn.axisgrid.PairGrid at 0x7f4907d10e90>
```

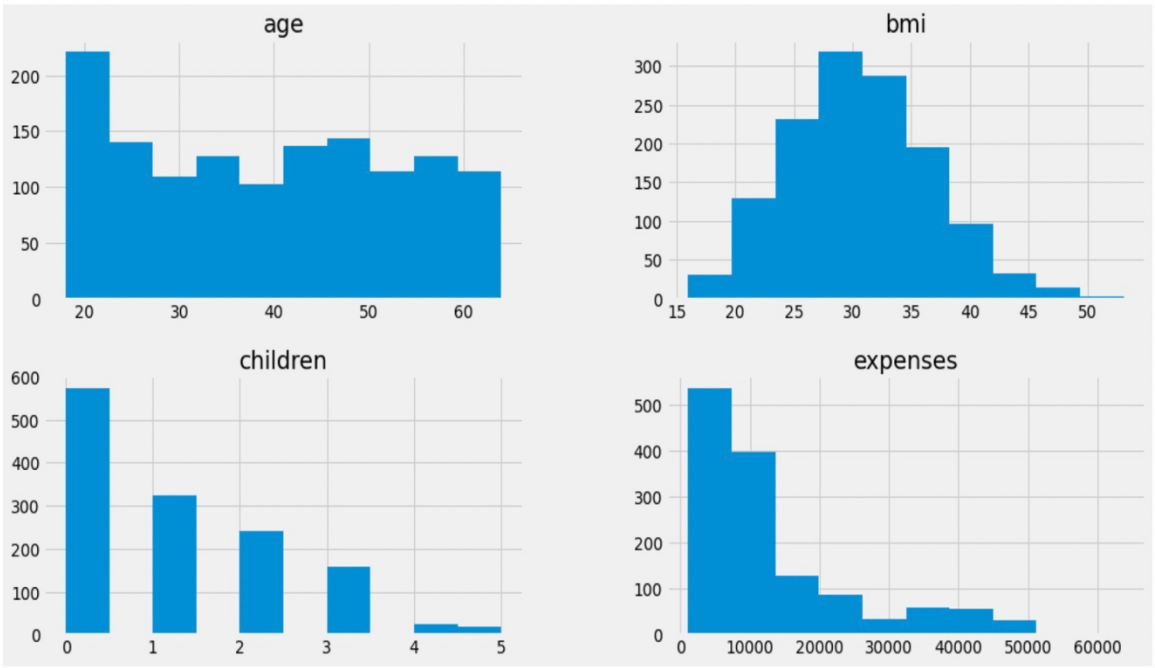


Visualizing data using Pair Plot



Correlation Matrix on Raw Data

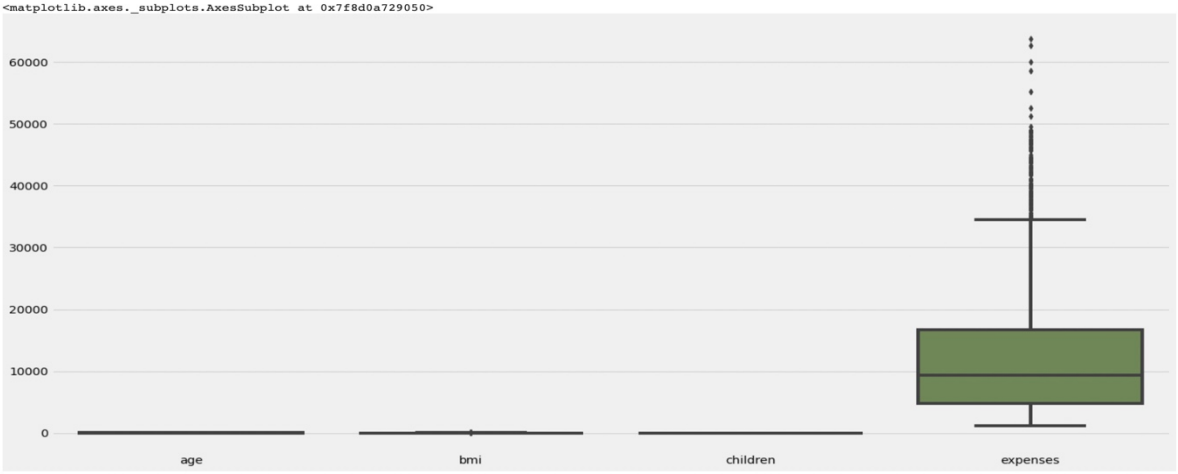
```
# Histogram per each numerical column

data.hist(figsize=(16, 8))

plt.savefig("Histogram.png")
```



Visualizing data using Histogram

```
# Boxplot of all numerical variables
sns.boxplot(data = data)
```

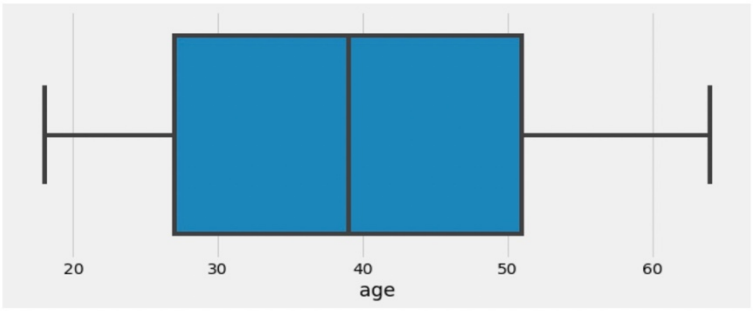
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8d0a729050>
```



Visualizing data using Box Plot on all numerical variables

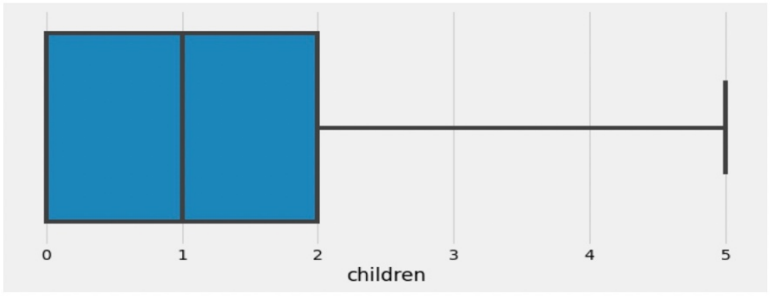
Box Plots of individual variables:

```
[18] # Boxplot of age column

     plt.figure(figsize=(10, 4))
     sns.boxplot(x=data['age'])
     plt.savefig("Boxplot1.png")
```



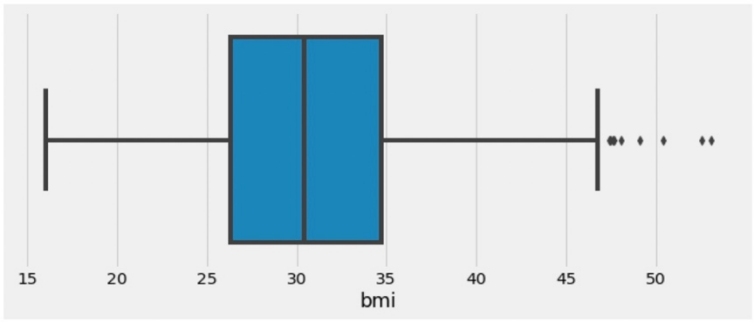
Visualizing data using Box Plot on age column

```
# Boxplot of children column

plt.figure(figsize=(10, 4))
sns.boxplot(x=data['children'])
plt.savefig("Boxplot3.png")
```



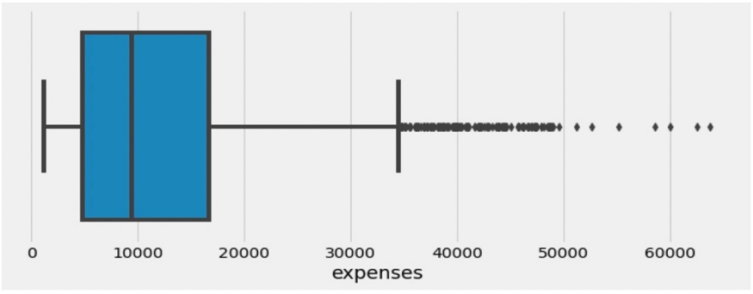
Visualizing data using Box Plot on children column

```
# Boxplot of bmi column

plt.figure(figsize=(10, 4))
sns.boxplot(x=data['bmi'])
plt.savefig("Boxplot2.png")
```



Visualizing data using Box Plot on bmi column

```
# Boxplot of expenses column

plt.figure(figsize=(10, 4))
sns.boxplot(x=data['expenses'])
plt.savefig("Boxplot4.png")
```

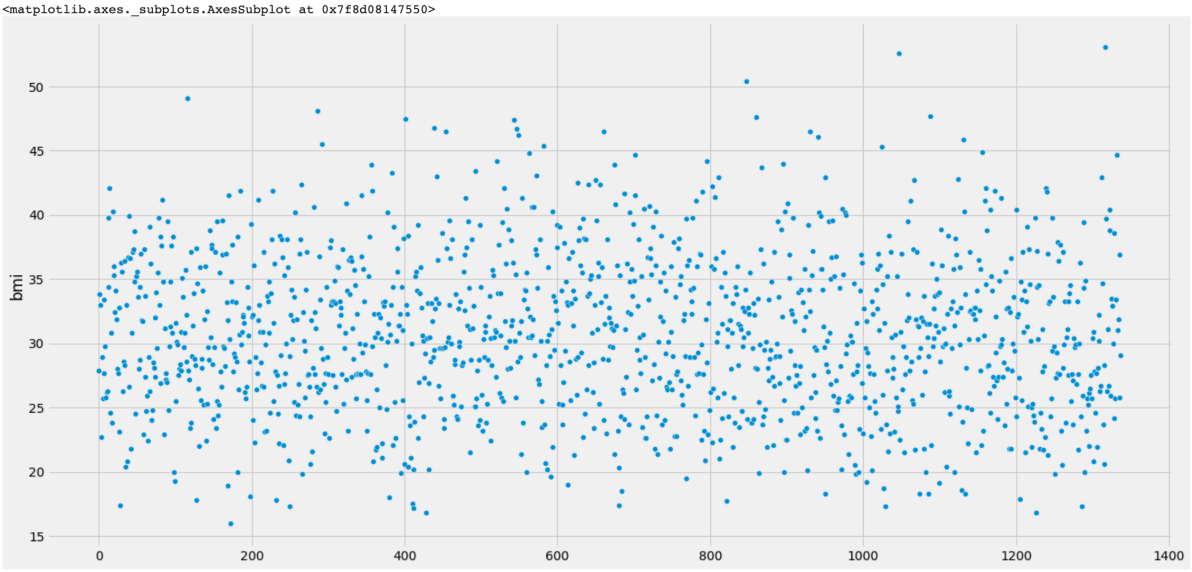


Visualizing data using Box Plot on expenses column

Using Scatter Plots to show outliers on bmi and expenses:

```
[ ] # Scatter plot for visualizing outliers on bmi

    sns.scatterplot(data = data['bmi'])
```

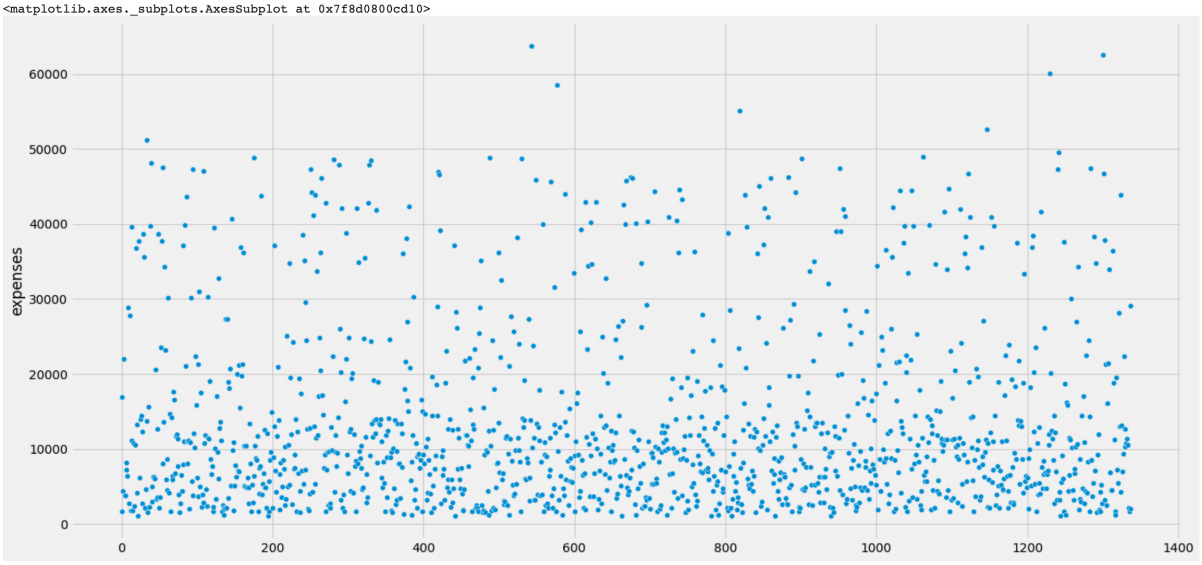
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8d08147550>
```



Visualizing outliers using Scatter Plot on bmi column

```
[ ] # Scatter plot for visualizing outliers on expenses

    sns.scatterplot(data = data['expenses'])
```

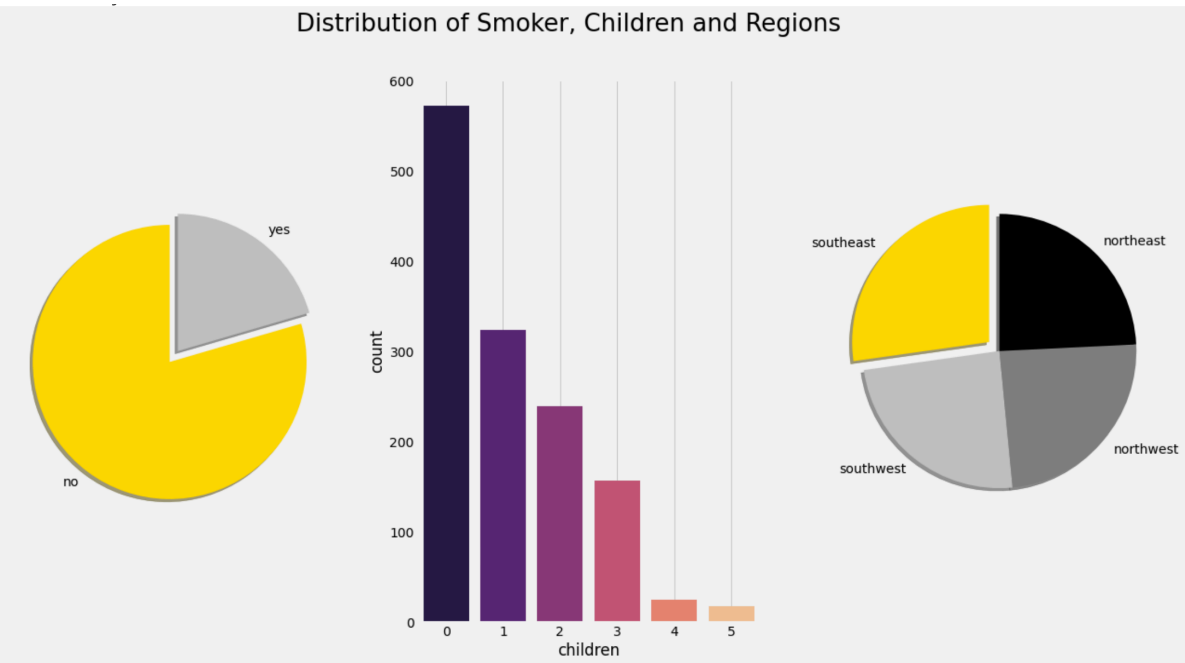
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8d0800cd10>
```



Visualizing outliers using Scatter Plot on expenses column

Univariate Analysis:

- Only one variable is involved.
- It's used to figure out how the variables in the dataset are distributed and to extract useful data from them.
- It can be used to examine both numerical and categorical variable distributions.



Visualizing distribution of Smoker, Children and Regions

Here we have used a pie chart to plot the Smoker Column, as the Smoker column has only two values: **Yes and No**. We have found 20.48% of the subjects are smokers and 79.52 % are non-smoker.

Using a Count plot, we have shown the subjects having children ranging from 0 to 5 and it has been computed and observed from the count plot also that those who are having no children are highest in number.

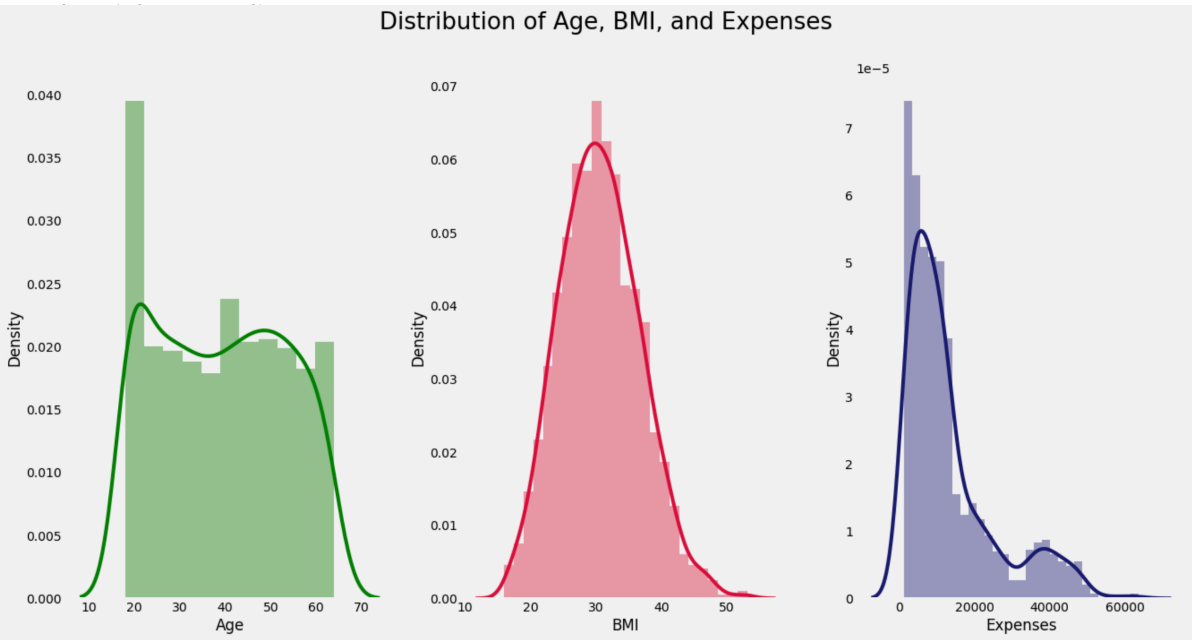
- Number of Subjects having no children- 574
- Number of Subjects having one child- 324
- Number of Subjects having two children- 240
- Number of Subjects having three children- 157
- Number of Subjects having four children- 25
- Number of Subjects having five children-18
- We have again used a pie chart to plot the number of inhabitants in the region column which consists of four segments: Northeast, Northwest, Southeast, Southwest

The number of Southwest and Northwest are the same and the value is 324, but the number of inhabitants in Northeast and Southeast are respectively 324 and 364.

Similarly, we plotted the distribution of Age, BMI and Expenses as below.

We have an equal number of people of all ages.

The BMI of the patients seems to be normally distributed where maximum people have BMI around 30 and very few people have less BMI around 10, similarly very few people have high BMI around 60.



Visualizing distribution of Age, BMI and Expenses

Data Processing:

- Check for the categorical columns.

```
[ ]  # Checking for the categorical columns

     data.select_dtypes('object').columns

     Index(['sex', 'smoker', 'region'], dtype='object')
```

- We used **One Hot Encoding** method and encoded the categorical variables “sex”, “region”, and “expenses” to convert them to numerical variables for executing the correlation b/w the features

```
[ ]  # Lets use One Hot Encoding to encode the above categorical columns

     one_hot_encoded_data = pd.get_dummies(data, columns = ['sex', 'smoker', 'region'])
     one_hot_encoded_data
```

	age	bmi	children	expenses	sex_female	sex_male	smoker_no	smoker_yes	region_northeast	region_northwest	region_southeast	region_southwest
0	19	27.9	0	16884.92	1	0	0	1	0	0	0	1
1	18	33.8	1	1725.55	0	1	1	0	0	0	1	0
2	28	33.0	3	4449.46	0	1	1	0	0	0	1	0
3	33	22.7	0	21984.47	0	1	1	0	0	1	0	0
4	32	28.9	0	3866.86	0	1	1	0	0	1	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...
1333	50	31.0	3	10600.55	0	1	1	0	0	1	0	0
1334	18	31.9	0	2205.98	1	0	1	0	1	0	0	0
1335	18	36.9	0	1629.83	1	0	1	0	0	0	1	0
1336	21	25.8	0	2007.95	1	0	1	0	0	0	0	1
1337	61	29.1	0	29141.36	1	0	0	1	0	1	0	0

1337 rows × 12 columns

- After encoding, we reordered the data by shifting target variable to end of the columns

```
[ ]  # Re-ordering the data by shifting Expenses column to end

     column_to_reorder = one_hot_encoded_data.pop('expenses')

     one_hot_encoded_data.insert(len(one_hot_encoded_data. columns), 'expenses', column_to_reorder)

     one_hot_encoded_data
```

	age	bmi	children	sex_female	sex_male	smoker_no	smoker_yes	region_northeast	region_northwest	region_southeast	region_southwest	expenses
0	19	27.9	0	1	0	0	1	0	0	0	1	16884.92
1	18	33.8	1	0	1	1	0	0	0	1	0	1725.55
2	28	33.0	3	0	1	1	0	0	0	1	0	4449.46
3	33	22.7	0	0	1	1	0	0	1	0	0	21984.47
4	32	28.9	0	0	1	1	0	0	1	0	0	3866.86
...	...	...	...	...	...	...	...	...	...	...	...	...
1333	50	31.0	3	0	1	1	0	0	1	0	0	10600.55
1334	18	31.9	0	1	0	1	0	1	0	0	0	2205.98
1335	18	36.9	0	1	0	1	0	0	0	1	0	1629.83
1336	21	25.8	0	1	0	1	0	0	0	0	1	2007.95
1337	61	29.1	0	1	0	0	1	0	1	0	0	29141.36

1337 rows × 12 columns

A Snippet of Data after encoding categorical variables

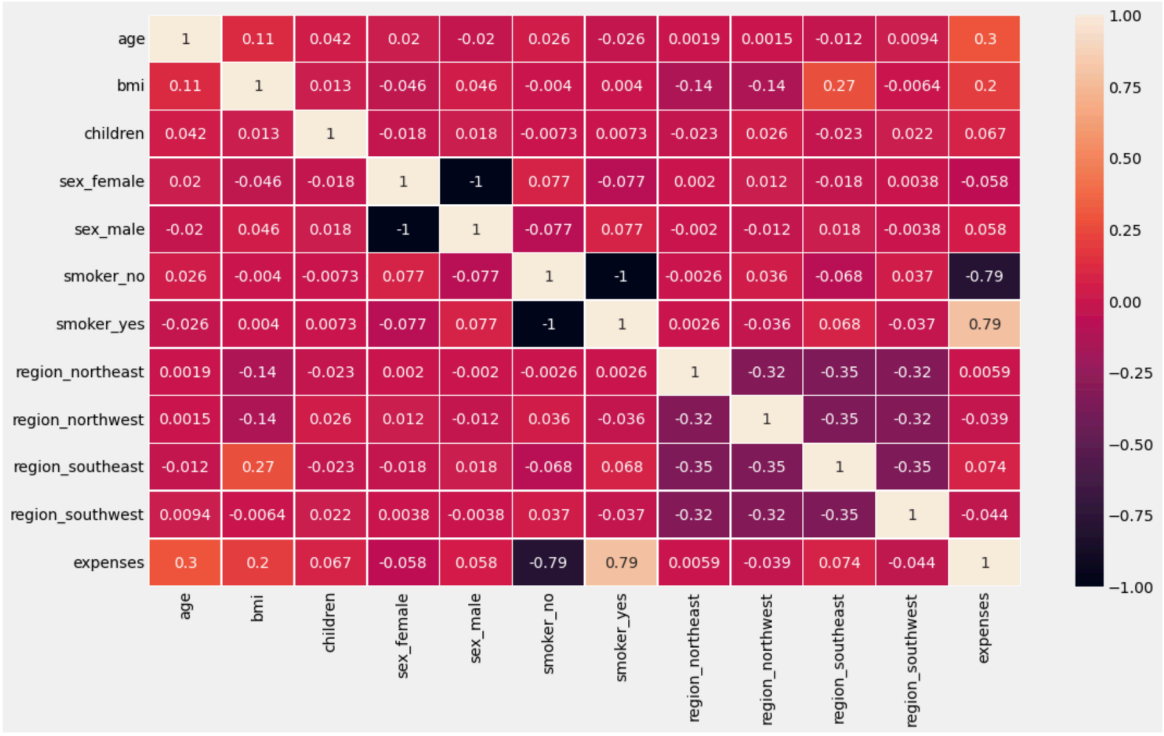
- Verifying the One Hot Encoding Process

```
[ ]  # Lets verify the One Hot Encoding process

     print('Columns in original data frame:\n',data.columns.values)
     print('\nNumber of rows and columns in the dataset:',data.shape)

     print('\nColumns in data frame after One Hot Encoding:\n',one_hot_encoded_data.columns.values)
     print('\nNumber of rows and columns in the dataset:',one_hot_encoded_data.shape)
```

```
Columns in original data frame:
 ['age' 'sex' 'bmi' 'children' 'smoker' 'region' 'expenses']

Number of rows and columns in the dataset: (1337, 7)

Columns in data frame after One Hot Encoding:
 ['age' 'bmi' 'children' 'sex_female' 'sex_male' 'smoker_no' 'smoker_yes'
 'region_northeast' 'region_northwest' 'region_southeast'
 'region_southwest' 'expenses']

Number of rows and columns in the dataset: (1337, 12)
```

- Extracting the encoded data to a csv file for reference

```
[ ]  # Lets extract the encoded data to a csv file for reference

     one_hot_encoded_data.to_csv("processed-insurance.csv")
```



Correlation Matrix on encoded data (Linear Correlation)

From the above representation we can say that,

- In between Age and Sex, we have very weak Correlation, age and BMI have weak correlation, age and smoker also have weak correlation. Age and Expenses have a moderate correlation, Age and South-West region also Age and North-west region have a negative Correlation.
- Sex and Expenses have a negative Correlation.
- Correlation between Children and Expenses is very weak.
- Correlation between Smoker and Expenses is strongly Negatively Correlated.
- Correlation between South-East region and Expenses is weak.
- Correlation between South-West region and Expenses is weakly negative.
- Correlation between North-West region and Expenses is weakly negative.
- Correlation between North-East region and Expenses is zero.

Data Partition:

- Let's form dependent and independent sets

```
[ ]  # Lets form dependent and independent sets

     X = one_hot_encoded_data.drop(['expenses'], axis = 1)
     y = one_hot_encoded_data['expenses']


     print(X.columns)
     print("\n")
     print(y.shape)
```

```
Index(['age', 'bmi', 'children', 'sex_female', 'sex_male', 'smoker_no',
       'smoker_yes', 'region_northeast', 'region_northwest',
       'region_southeast', 'region_southwest'],
      dtype='object')


(1337,)
```

- For the modeling, we split the dataset into train and test using train_test_split().

```
[ ]  # Lets perform Train - Test - Split

     from sklearn.model_selection import train_test_split

     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)

     print(X_train.shape)
     print(X_test.shape)
     print(y_train.shape)
     print(y_test.shape)
```

```
(1069, 11)
(268, 11)
(1069,)
(268,)
```

We have split the data into 80 : 20

Feature Scaling:

- Feature scaling is a technique for normalizing a set of independent variables or data features.
- Data Normalization is another name for feature scaling.
- It helps in the normalization of data within a particular range.
- It also helps in the speeding of algorithmic calculations.

- We performed Standardization using Standard Scaling by using StandardScaler() on the data before building the model as the values for a few features were exorbitantly high.

```
[ ]  # Lets perform Standardization using Standard Scaling

     from sklearn.preprocessing import StandardScaler

     scaler = StandardScaler()
     scaler.fit(X_train)
     X_train_scaled = scaler.transform(X_train)
     X_test_scaled = scaler.transform(X_test)
```

Normalizing the data using Standard Scaler Technique

Data Modeling

The first few attempts at model building for this project were performed using regression models from several libraries.

The models were:

- Linear Regression Model
- K-Nearest Neighbors Model
- Random Forest Model
- Ensemble of all above three models
- Weighted Average of all three models
- Cross Validation of all three models
- Comparing performance of all models

Evaluation Metrics:

- **R^2 Score**
    - It is generally used to determine the strength of correlation between the independent features and the target column.
- **Root Mean Squared Error(RMSE)**
    - It is the square root of the mean of the differences between actual and the predicted values.

Data used for the model building was divided with 80% used for training and the remaining 20% for testing. The models were then fitted using the training data set and predictions made on the test data set. Accuracy using the R^2 scoring metric was recorded.

Linear Regression

- A linear approach to modeling the relationship between a dependent and one or more independent variables is known as linear regression.
- The dependent and independent variables must have a linear relationship.
- Two variables must have trend lines that are either increasing or decreasing.
- In statistical terminology, all variables must have the same variance.

As our target variable is continuous dependent variable, Linear Regression is the best choice for predicting the future medical expenses.

- Here we predicted future medical expenses using Linear Regression on unscaled data and scaled data.
- We computed R^2 and RMSE values obtained as **0.7530** and 6445.47 for unscaled data.

Code Snippet for the Linear Regression model depicting the RMSE and R^2 values

```
[266] # Creating a simple Linear Regression Model

      from sklearn.linear_model import LinearRegression

      lin = LinearRegression()
      lin.fit(X_train,y_train)

      y_pred1 = lin.predict (X_test)

      print("Training Score     :", lin.score(X_train,y_train))
      print("Testing Score      :", lin.score(X_test,y_test))
```

```
Training Score     : 0.7488769969730097
Testing Score      : 0.7530509541914574
```

```
[267] # Model Accuracy -- R2 // RMSE

      from sklearn.metrics import r2_score, mean_squared_error


      mse = mean_squared_error (y_test, y_pred1)
      rmse = np.sqrt(mse)
      print("RMSE Score     :", rmse)

      r2_score = r2_score (y_test, y_pred1)
      print("R2 Score       :", r2_score)
```

```
RMSE Score     : 6445.473682867912
R2 Score       : 0.7530509541914574
```

Linear Regression Model – Unscaled Data

- For scaled data we got **0.7534** and 6440.49 as R^2 and RMSE values.

Code Snippet for the Linear Regression model depicting the RMSE and R^2 values

```
# Creating a simple Linear Regression Model with Scaled data

from sklearn.linear_model import LinearRegression

lin = LinearRegression()
lin.fit(X_train_scaled,y_train)

y_pred1_scaled = lin.predict (X_test_scaled)

print("Training Score     :", lin.score(X_train_scaled,y_train))
print("Testing Score      :", lin.score(X_test_scaled,y_test))
```

```
Training Score     : 0.7488441168659459
Testing Score      : 0.7534323358450128
```

```
[269] # Model Accuracy for scaled data -- R2 // RMSE

from sklearn.metrics import r2_score, mean_squared_error


mse = mean_squared_error (y_test, y_pred1_scaled)
rmse = np.sqrt(mse)
print("RMSE Score     :", rmse)

r2_score = r2_score (y_test, y_pred1_scaled)
print("R2 Score       :", r2_score)
```

```
RMSE Score     : 6440.494649185112
R2 Score       : 0.7534323358450128
```

Linear Regression Model – Scaled Data

We observed that for scaled data, the Accuracy has been improved.

K-Nearest Neighbors Model

- K-NN is a simple algorithm that stores all available cases and predicts the numerical target variable base on similarity measure.
- This means that the new point is assigned a value based on how closely it resembles the values in the training set.

- Here we predicted future medical expenses using K-NN model on unscaled data and scaled data.
- We computed R^2 and RMSE values obtained as **0.3391** and 10543.63 for unscaled data.

Code Snippet for the K-NN model depicting the RMSE and R^2 values

```
[270] # Creating a simple KNN Model

      from sklearn.neighbors import KNeighborsRegressor
      knn = KNeighborsRegressor()
      knn.fit(X_train,y_train)

      y_pred2 = knn.predict (X_test)

      print("Training Score     :", knn.score(X_train, y_train))
      print("Testing Score      :", knn.score(X_test, y_test))
```

```
Training Score     : 0.4954577380856632
Testing Score      : 0.3391870610379586
```

```
[271] # Model Accuracy -- R2 // RMSE

      from sklearn.metrics import r2_score, mean_squared_error


      mse = mean_squared_error (y_test, y_pred2)
      rmse = np.sqrt(mse)
      print("RMSE Score     :", rmse)

      r2_score = r2_score (y_test, y_pred2)
      print("R2 Score       :", r2_score)
```

```
RMSE Score     : 10543.636680643014
R2 Score       : 0.3391870610379586
```

K-NN Model – Unscaled Data

- For scaled data we got **0.7982** and 5825.76 as R^2 and RMSE values.

Code Snippet for the KNN model depicting the RMSE and R^2 values

```
[330] # Creating a simple KNN Model with scaled data

      from sklearn.neighbors import KNeighborsRegressor
      knn = KNeighborsRegressor()
      knn.fit(X_train_scaled,y_train)

      y_pred2_scaled = knn.predict (X_test_scaled)

      print("Training Score     :", knn.score(X_train_scaled, y_train))
      print("Testing Score      :", knn.score(X_test_scaled, y_test))

      Training Score     : 0.8636905165020783
      Testing Score      : 0.798254768782863
```

```
[331] # Model Accuracy for scaled data -- R2 // RMSE

      from sklearn.metrics import r2_score, mean_squared_error


      mse = mean_squared_error (y_test, y_pred2_scaled)
      rmse = np.sqrt(mse)
      print("RMSE Score     :", rmse)

      r2_score = r2_score (y_test, y_pred2_scaled)
      print("R2 Score       :", r2_score)

      RMSE Score     : 5825.762855660883
      R2 Score       : 0.798254768782863
```

K-NN Model – Scaled Data

We observed that for scaled data, the Accuracy has been improved drastically.

This proves that **K-NN model is best when working with scaled data.**

Random Forest Model

- Random forest is an ensemble learning method for classification and regression by constructing multiple number of decision trees at training time.
- And it outputs the average prediction of the individual trees in case of regression and mode of the classes in case of classification.
- It is one of the Most powerful Machine Learning algorithms which works well in most cases.

- Here we predicted future medical expenses using Random Forest on unscaled data and scaled data.
- We computed R^2 and RMSE values obtained as **0.8375** and 5226.96 for unscaled data.

Code Snippet for the Random Forest model depicting the RMSE and R^2 values

```
# Creating a Random Forest Model

from sklearn.ensemble import RandomForestRegressor

rf = RandomForestRegressor()
rf.fit (X_train, y_train)

y_pred3 = rf.predict (X_test)

print("Training Score     :", rf.score(X_train, y_train))
print("Testing Score      :", rf.score(X_test, y_test))
```

```
Training Score     : 0.9762764499033985
Testing Score      : 0.837595785661922
```

```
[333] # Model Accuracy -- R2 // RMSE

from sklearn.metrics import r2_score, mean_squared_error

mse = mean_squared_error (y_test, y_pred3)
rmse = np.sqrt(mse)
print("RMSE Score     :", rmse)

r2_score = r2_score (y_test, y_pred3)
print("R2 Score       :", r2_score)
```

```
RMSE Score     : 5226.967631772907
R2 Score       : 0.837595785661922
```

Random Forest Model – Unscaled Data

- For scaled data we got **0.8368** and 5238.56 as R^2 and RMSE values.

Code Snippet for the Random Forest model depicting the RMSE and R^2 values

```
[334] # Creating a Random Forest Model with scaled data

      from sklearn.ensemble import RandomForestRegressor

      rf = RandomForestRegressor()
      rf.fit (X_train_scaled, y_train)

      y_pred3_scaled = rf.predict (X_test_scaled)

      print("Training Score     :", rf.score(X_train_scaled, y_train))
      print("Testing Score      :", rf.score(X_test_scaled, y_test))

      Training Score     : 0.9759904982575159
      Testing Score      : 0.8368746225543691

      # Model Accuracy for scaled data -- R2 // RMSE

      from sklearn.metrics import r2_score, mean_squared_error


      mse = mean_squared_error (y_test, y_pred3_scaled)
      rmse = np.sqrt(mse)
      print("RMSE Score     :", rmse)

      r2_score = r2_score (y_test, y_pred3_scaled)
      print("R2 Score       :", r2_score)

      RMSE Score     : 5238.560067496894
      R2 Score       : 0.8368746225543691
```

Random Forest Model – Scaled Data

Ensemble of all above three models

- Ensemble models is a machine learning approach to combine multiple other models in the prediction process.
- Those models are referred to as base estimators.
- It is a solution to overcome the technical challenges of building a single estimator

- Here we predicted future medical expenses using Ensemble by average of all models on unscaled data and scaled data.
- We computed R^2 and RMSE values obtained as **0.7617** and 6330.46 for unscaled data.

Code Snippet for Ensemble of all models depicting the RMSE and R^2 values

```
[336] # Trying to identify a good ensemble of methods

      # Ensemble by average of all three models

      from sklearn.metrics import r2_score, mean_squared_error

      avg_model = (y_pred1 + y_pred2 + y_pred3) / 3


      mse = mean_squared_error (y_test, avg_model)
      rmse = np.sqrt(mse)
      print("RMSE Score     :", rmse)

      r2_score = r2_score (y_test, avg_model)
      print("R2 Score       :", r2_score)

      RMSE Score     : 6330.466538209766
      R2 Score       : 0.7617849979933369
```

Ensemble by Average – Unscaled Data

- For scaled data we got **0.8266** and 5399.65 as R^2 and RMSE values.

Code Snippet for Ensemble of all models depicting the RMSE and R^2 values

```
[337] # Ensemble by average for scaled models

      from sklearn.metrics import r2_score, mean_squared_error

      avg_model = (y_pred1_scaled + y_pred2_scaled + y_pred3_scaled) / 3


      mse = mean_squared_error (y_test, avg_model)
      rmse = np.sqrt(mse)
      print("RMSE Score     :", rmse)

      r2_score = r2_score (y_test, avg_model)
      print("R2 Score       :", r2_score)

      RMSE Score     : 5399.657692747645
      R2 Score       : 0.8266874024911137
```

Ensemble by Average – Scaled Data

Weighted Average of all three models

- We created a weighted average of all three models with below weightage:
  - Random Forest – 50%
  - KNN – 30%
  - Linear Regression – 20%

```
[343] # Creating a weighted average model

      # Giving 50% weight to random forest
      # 30% weight to knn
      # and 20% weight to linear regression

      weight_avg_model = 0.2*y_pred1_scaled + 0.3*y_pred2_scaled + 0.5*y_pred3_scaled

      # Checking the Model accuracy
      from sklearn.metrics import r2_score, mean_squared_error

      mse = mean_squared_error(y_test, weight_avg_model)
      rmse = np.sqrt(mse)
      print("RMSE Score     :", rmse)

      r2_score = r2_score(y_test, weight_avg_model)
      print("R2 Score       :",r2_score)

      RMSE Score     : 5243.92529493142
      R2 Score       : 0.8365403120100317
```

Cross Validation – 5 Fold

- Cross Validation is a resampling procedure which is used to evaluate the machine learning models on limited data samples.
- The goal of cross validation is to test the model's ability to predict new data that was not used in estimating it.
- It has a single parameter called **k**, which indicates the number of groups that the data would be split into.
- Here, we set the value of k = 5.

```
[342] # 5-fold Cross validation

      from sklearn.model_selection import cross_val_score
      scores = cross_val_score(rf, X, y, cv= 5)
      print("The 5-fold Cross Validation scores using Random Forest model   :",scores)

      The 5-fold Cross Validation scores using Random Forest model   : [0.85036998 0.77562962 0.86951228 0.83052304 0.85310877]
```

The 5-fold Cross Validation scores using Random Forest model:

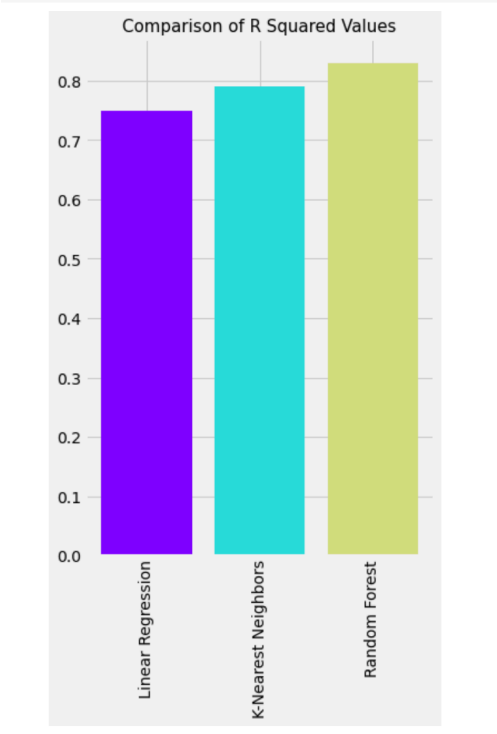
[0.85036998 0.77562962 0.86951228 0.83052304 0.85310877]

Comparison of all three models

- We have created a NumPy array of the R^2 score of all three models, Linear Regression, Random Forest, and K-NN.
- An array for the labels was also created as well, to compare these Values using bar charts.
- Here a Rainbow palette has been used and the Bar plot built using the seaborn Library shows a higher R^2 score value for Random Forest and lowest for Linear Regression.
- That means, the **Random Forest Model is the Best Choice** whereas the Linear Regression Model is the worst for this Case.
- We have successfully built our predictive model and compared these predictive models based on their accuracies and Results.

```
[ ]  # Comparing R Squared Values

     r2_score = np.array([0.75, 0.79, 0.83])
     labels = np.array(['Linear Regression', 'K-Nearest Neighbors' , 'Random Forest'])
     index = np.argsort(r2_score)
     color = plt.cm.rainbow(np.linspace(0, 1, 4))

     plt.style.use('fivethirtyeight')
     plt.rcParams['figure.figsize'] = (5, 8)

     plt.bar(range(len(index)), r2_score[index], color = color)
     plt.xticks(range(0, 3), ['Linear Regression', 'K-Nearest Neighbors','Random Forest'], rotation = 90)
     plt.title('Comparison of R Squared Values', fontsize = 15)
     plt.show()
```

Below is the bar plot to show the performance of the three used models.



Evaluation

- We came to know that the Most Important Factor to Predict the Medical Expenses of a subject is Smoking Behavior and Age, that means, smoking is Bad for Health, as we already know that and which inevitably increases medical expenses as due to smoking one is likely to fall ill more than the nonsmokers.

- We also found that with increasing of age, one needs to take some more care and precautions for your health as with the increase of age health becomes fragile, so they go for frequent medical check-up, likely to fall ill quickly as with the increase of age immunity falls so they adopt measures to stay healthy by taking medicines and engaging in some physical activities.

- Apart from that we also understood that Gender, Number of Children, the Region also have a good impact on determining Medical Expenses.

Conclusion

We have built three models among which the **Random Forest Regressor model** shows the best result through which we can say **83.75%** variability of expenses can well be explained by predictor variables and which yields comparatively low RMSE value so our predicted expense through this model will not vary too much from the actual expense.

Recommendations

The accuracy can be much more enhanced with the following recommendations:

- The data sample size is relatively small in this case, especially with a perspective of applying this in real world.

- The number of features is also limited in this dataset. With an addition of 5 or more valid/logical features, the predictions can be improved much further.

- We can try converting the Expense column to a normal distribution using log or square root transformation for removing the skewness.

- Future enhancements to this project could include experiments with additional predictive models such as Gradient Boosting or SVM models. Improved accuracy of the linear regression models is likely with additional preprocessing of the data to better conform to the assumptions of linear modeling.

Future Work

- We plan to take this project further as health is center of everyone's life and every part of our life relies on good health.

- We intend to develop an application to predict possible medical costs using PyCharm editor on AWS platform by plugging in basic details such as age, sex, smoker, children, region and then pressing a submit button.

- This application will help people understand the factors that are making them unfit so that they can reduce their medical expenses.

- We can also use this health expense predictor in hospitals so that patients can adjust their medical expenses.