Mgbemena Mmesomachukwu Chukwuemeka

2315192

МИВТ-23-6А

MEDICAL HEALTH INSURANCE PREDICTION USING A LINEAR REGRESSION MODEL IMPLEMENTED IN PYTHON

ABOUT THE PROJECT

This project delves into predicting medical insurance costs using machine learning. By analyzing factors like age, BMI, smoking habits, etc., we aim to unravel the cost determinants. Our focus is on data-driven insights for stakeholders, fostering informed decision-making in healthcare economics. Through analysis and linear regression model, the aim is to achieve precise predictions.

METHODOLOGY

1. Data Collection:
    - Acquire a diverse dataset encompassing key variables such as age, BMI, children, smoker status, region and the insurance costs.
2. Data Analysis:
    - Conduct exploratory data analysis (EDA) to unveil patterns, patterns, distributions, providing insights into potential predictors.
3. Data Preprocessing:
    - Address data quality issues by handling missing values, outliers, and encoding categorical variables.
4. Train/Test Split:
    - Split the dataset into training and testing sets to facilitate model training and evaluation. The typical split is around 80% for training and 20% for testing.
5. Linear Regression Model:

- Implement a linear regression model making use of the features and the target variable.

6. Model Evaluation:
    - Evaluate the performance of the model using appropriate metrics. Common metrics for regression problems include Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) and R-squared.
7. Conclusion

SETTING UP THE ENVIRONMENT

In line with project, I made use of the Python programming language and imported the following python libraries:

1. NumPy: for facilitating mathematical and logical operations on arrays.
2. Pandas: for simplifying data manipulation, cleaning and exploration.
3. Matplotlib: enables the creation of interactive visualizations.
4. Seaborn: eases the process of interactive visualization creation.
5. Scikit-Learn: provides a set of tools for machine learning tasks.

DATA COLLECTION

The data was collated from Kaggle Datasets on medical insurance in the United States (US).

Factors that affect the cost of medical insurance include:

1. Age: Age of primary beneficiary
2. Sex: The gender of the insurance benefactor- male or female
3. BMI: The Body Mass Index (BMI) gives an understanding of the human and the relationship of the weight of the body to the height, kg/m^2
4. Children: Number of children covered by the medical health insurance
5. Smoker: Smoking status of the individual
6. Region: The residential area of the benefactor in the US

DATA ANALYSIS

1. **Importing the dataset:** To import the dataset, we make use of Pandas method.
```
insurance_df = pd.read_csv('insurance.csv')
```

2. **Checking the dataset head:** Retrieving the top five DataFrame is a common practice to see a quick overview of the data. It is particularly useful for initial data inspection and ensuring that the dataset contains the expected information.
```
insurance_df.head()
```

Output:

	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

3. **Describe method:** The describe method is used to gain insights into the numerical aspects of the DataFrame. When applied, it provides a summary of key statistical measures for each numerical column in the dataset. This includes count, mean, standard deviation, minimum, 25th percentile, median (50th percentile), 75th percentile, and maximum values. The describe() method is valuable for a quick overview of the central tendency, dispersion, and distribution of the numerical data within the DataFrame.

```
insurance_df.describe()
```

Output:

	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

4. **Info method:** the info() method in pandas is a handy function to obtain essential information about a DataFrame. When invoked, it provides a concise summary,

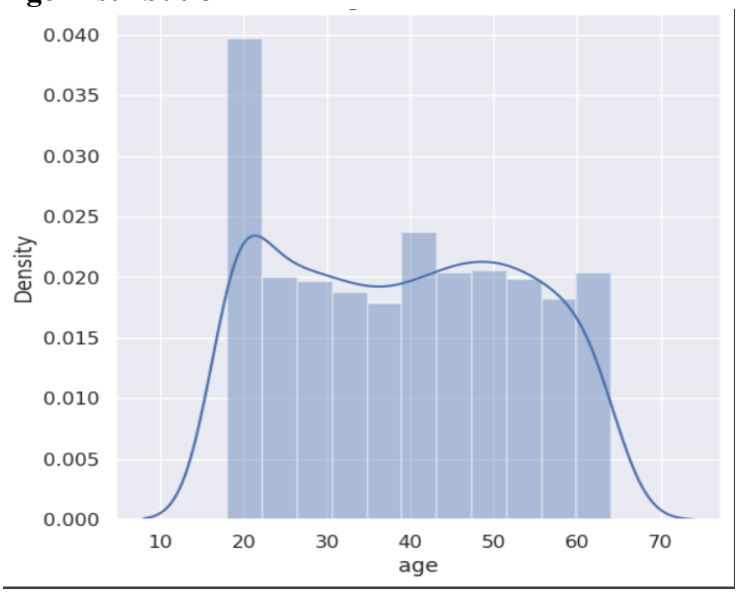
including the data types of each column, the number of non-null values, and the memory usage. This is valuable for understanding the structure of the dataset, identifying missing values, and ensuring that the data types align with the intended analysis.

df.info()

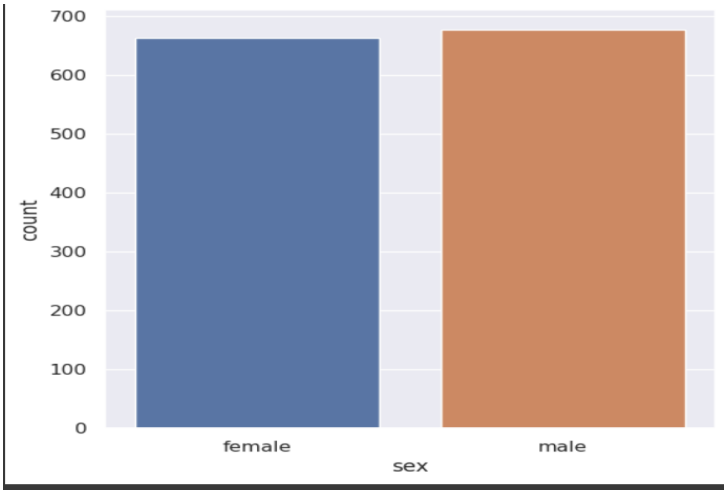
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
#   Column    Non-Null Count  Dtype
---  ------    --------------  -----
0   age       1338 non-null   int64
1   sex       1338 non-null   object
2   bmi       1338 non-null   float64
3   children  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)
```

Using the visualization libraries, Matplotlib and Seaborn, the following graphs were plotted:

1. **Age Distribution**

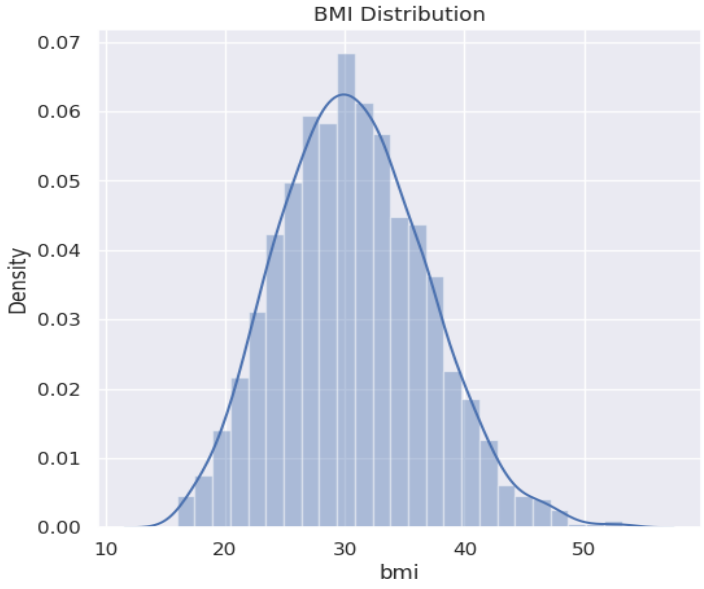


2. **Sex Distribution**

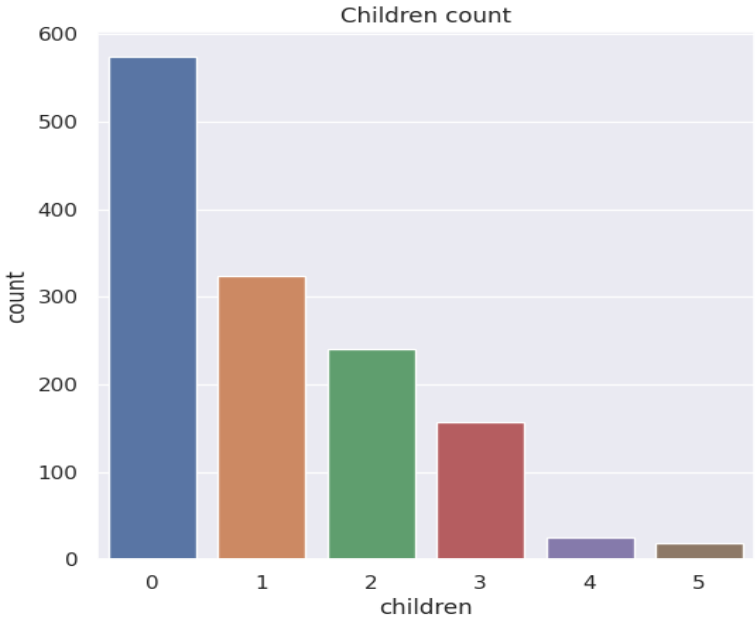


Male – 676
Female – 662

3. **BMI Distribution**



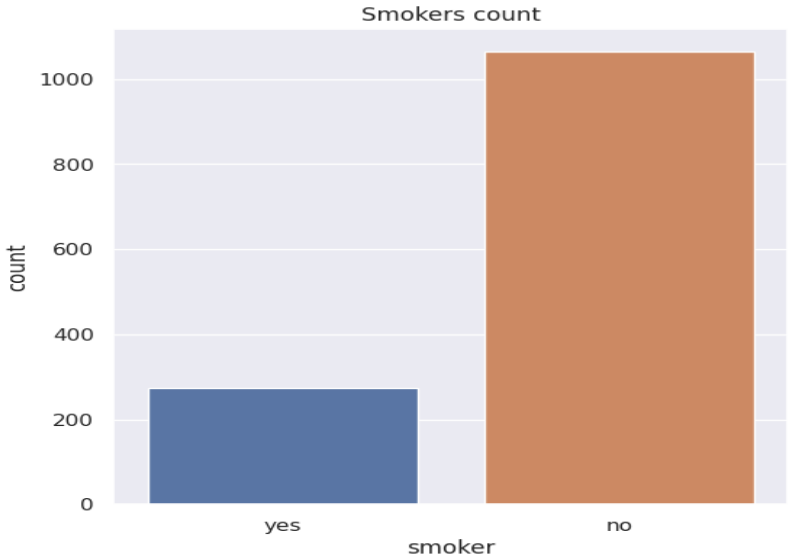
4. **Children Count**



insurance_df['children'].value_counts()

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

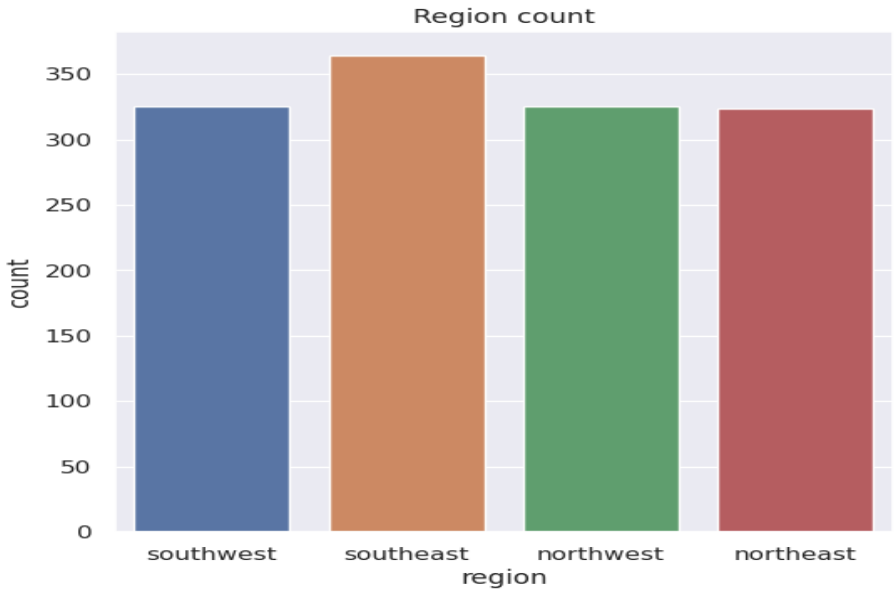
5. **Smoker Count**



insurance_df['smoker'].value_counts()

Output:
```
no      1064
yes      274
Name: smoker, dtype: int64
```

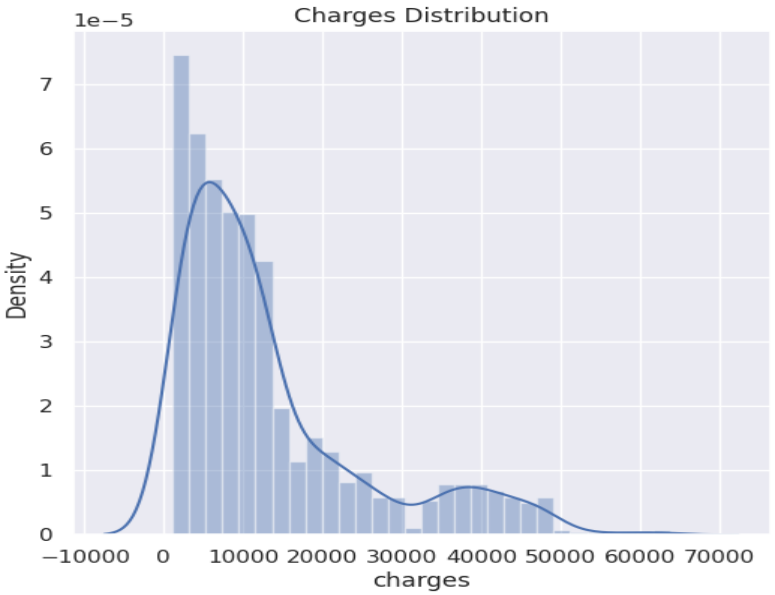
6. **Region Count**



insurance_df['region'].value_counts()

Output:
```
southeast    364
southwest    325
northwest    325
northeast    324
Name: region, dtype: int64
```

7. **Charges Distribution**



DATA PREPROCESSING

Data preprocessing is a crucial step in machine learning that involves transforming raw data into a suitable format for training models. This phase aims to enhance quality and reliability in the input data, ensuring optimal performance of the machine learning algorithms.

To check for missing values, using Pandas:

insurance_df.isnull().sum()

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

The categorical features in the dataset is as follows:

1. Sex
2. Smoker
3. Region

Encoding categorical features is a fundamental aspect of data preprocessing in machine learning, where non-numeric variables, such as categorical labels or text, are converted into a numerical format. This transformation is essential because most machine learning algorithms rely on numerical input. Using label encoding, the categorical features were converted to numerical values, thus, allowing this data to be processed.

```
# Encoding sex column
insurance_df.replace({'sex':{'male':0, 'female':1}}, inplace=True)

# Encoding smoker column
insurance_df.replace({'smoker':{'no':0, 'yes':1}}, inplace=True)

# Encoding region column
insurance_df.replace({'region':{'southeast':0, 'southwest':1,
'northeast':2, 'northwest':3}}, inplace=True)
```

```
insurance_df.head()
```

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

Next, the features and the target variable are split such that variable X contains the features and variable Y contains the target.

```
X = insurance_df.drop(columns='charges', axis=1)
Y = insurance_df['charges']
```

TRAIN/TEST SPLIT

In this process, the available dataset is divided into two subsets: the training set and the testing set.

1. **Training Set:**
    - The larger portion of the dataset used to train the machine learning model.
    - The model learns patterns, relationships, and features from this data.
2. **Testing Set:**
    - A smaller, independent subset of the data that the model has not seen during training.
    - Used to evaluate the model's performance and generalization to unseen data.

By splitting the data into these two sets, machine learning practitioners can gauge how well their model can make predictions on new, unseen data. 80% of the dataset will serve as training data, whilst 20% as test data. Using sklearn library, this simplifies the process. Random state is used to control the randomness, so that it is reproducible.

```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size=0.2,
random_state=2)
```

MODEL: LINEAR REGRESSION

Using the Scikit-Learn library, we instantiate the LinearRegression() method and fit the data to the model.

```
linear_regression = LinearRegression()
linear_regression.fit(X_train, Y_train)
```

MODEL EVALUATION

1. **Mean Squared Error (MSE):**
    - MSE measures the average squared difference between predicted and actual values.
    - Provided MSE: 38337035.49
    - A lower MSE indicates better model performance, as it signifies smaller errors in predictions. In your case, the MSE suggests that, on average, your model's predictions are off by approximately 38,337,035.49 squared units.
2. **Mean Absolute Error (MAE):**
    - MAE computes the average absolute differences between predicted and actual values.
    - Provided MAE: 4267.21

- To better interpret, compare this value with the baseline MAE (9312.88). A lower MAE indicates improved model accuracy. In your case, your model's predictions, on average, deviate by approximately 4,267.21 units, which is substantially better than the baseline MAE.

3. **Baseline MAE:**
    - Baseline MAE is the mean absolute error of a simple baseline model, often based on the mean or median of the target variable.
    - Provided Baseline MAE: 9312.88
    - Your model's MAE is significantly lower than the baseline MAE, suggesting that your linear regression model is providing more accurate predictions than a simplistic baseline approach.
4. **R-squared (Coefficient of Determination):**
    - R-squared assesses the proportion of the variance in the dependent variable that is predictable from the independent variables.
    - R-squared on Test Data: 0.7515
    - R-squared on Train Data: 0.7447
    - R-squared values range from 0 to 1, with 1 indicating a perfect fit. In your case, the R-squared values suggest that approximately 75.15% of the variance in the test data and 74.47% in the training data is explained by your linear regression model. These values indicate a reasonably good fit, capturing a substantial portion of the data's variability.

CONCLUSION

In conclusion, the linear regression model performs well, as indicated by lower MSE and MAE compared to the baseline, along with reasonably high R-squared values. These metrics the affirm the model’s ability to make accurate predictions and explain significant proportion of the variance in the target variable.

Using the model that has been trained, we can make a predictive system that takes in data(features) and gives a health insurance cost.