MEDICAL INSURANCE COST PREDICTION USING MACHINE LEARNING ALGORITHM

TEAM MEMBERS

ARUN PRAKASH -20140102

MOHAMMED KAIF -20140103

ANBU SELVAN -20140111

INTRODUCTION :

Medical insurance cost prediction is the process of estimating the potential expenses associated with providing health insurance coverage to individuals or groups. It involves analyzing various factors that can influence the cost of insurance, such as age, health status, coverage level, location, lifestyle factors, family size, insurance plan type, and the insurance company.

The first step in the process is collecting relevant data from various sources, such as insurance claim data, medical records, and demographic information. This data is then preprocessed, which may involve cleaning, organizing, and transforming the data into a suitable format for analysis.

Next, predictive models are developed using statistical or machine learning techniques. These models use the collected data and selected features to estimate the potential medical insurance costs. Common modelling techniques include linear regression, decision trees, neural networks, and other approaches depending on the complexity of the data and desired accuracy of the predictions.

Once the models are developed, they are evaluated using appropriate metrics to assess their accuracy, reliability, and generalization performance. This may involve cross-validation, testing against a holdout dataset, or other validation methods. The models may also be refined or optimized based on the evaluation results.

Interpretation and validation of the results are then conducted to ensure that the predictions are meaningful and reliable. This may involve comparing the predicted insurance costs with actual insurance costs from historical data or validating the results with domain experts.

Finally, once the model is validated and approved, it can be deployed in a real-world setting, such as an insurance company's pricing system or a web-based tool, to predict medical insurance costs for new cases.

It's important to note that actual medical insurance costs may vary depending on individual circumstances, and consulting with qualified professionals and considering multiple factors is essential for accurate and up-to-date medical insurance cost estimation. The process of medical insurance cost prediction requires careful data analysis, model development, validation, and interpretation to ensure reliable and meaningful results.

DATA COLLECTION AND PREPROCESSING :

1. Data collection: In this step, data is collected from various sources that are relevant to estimating medical insurance costs. This may include insurance claim data, medical records, demographic information, and other relevant data points. The data may include information on factors such as age, health status, coverage level, location, lifestyle factors, family size, insurance plan type, and historical insurance cost data. The data can be obtained from internal sources within an insurance company or from external sources such as public databases or third-party data providers.
2. Data preprocessing: Once the data is collected, it needs to be preprocessed to ensure that it is clean, organized, and ready for analysis. This step may involve several tasks, including:

a. Data cleaning: This involves identifying and handling any missing, inconsistent, or erroneous data. Missing data may be imputed using various techniques such as mean imputation, median imputation, or machine learning-based imputation methods. Inconsistent or erroneous data may be corrected or removed from the dataset.

b. Data integration: If data is collected from multiple sources, it may need to be integrated into a single dataset. This may involve matching and merging data based on common identifiers, such as patient ID or policy number.

c. Data transformation: Data may need to be transformed into a suitable format for analysis. This may involve converting categorical variables into numerical representations, normalizing numerical variables, or encoding categorical variables using techniques such as one-hot encoding or label encoding.

d. Feature selection: Relevant features or variables that are most likely to influence medical insurance costs are selected from the collected data. This step involves identifying and retaining the most important features while discarding irrelevant or redundant features, which helps to reduce the dimensionality of the data and focus on the most important factors.

e. Data validation: Data is checked for accuracy and consistency to ensure that it is reliable for analysis. This may involve performing data validation checks, such as range checks, consistency checks, or outlier detection.

Data collection and preprocessing are critical for the accuracy and reliability of the predictive models developed for medical insurance cost prediction. Proper data collection and preprocessing ensure that the data used for analysis is clean, organized, and appropriate for modeling, which helps to generate meaningful and reliable predictions of medical insurance costs.

Data Collection & Analysis

```
[ ] # loading the data from csv file to a Pandas DataFrame
    insurance_dataset = pd.read_csv('/content/insurance.csv')
```

```
    # first 5 rows of the dataframe
    insurance_dataset.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

```
[ ] # number of rows and columns
    insurance_dataset.shape
```

```
(1338, 7)
```

```
[ ]  # getting some informations about the dataset
     insurance_dataset.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
```

Categorical Features:

Sex

Smoker

Region

```
[ ]  # checking for missing values
     insurance_dataset.isnull().sum()
```

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

DATA ANALYSIS :

Data analysis is a crucial step in the process of medical insurance cost prediction. Once the data has been collected and preprocessed, it is analyzed to uncover patterns, relationships, and insights that can help in estimating medical insurance costs. Data analysis involves various techniques and methods to extract meaningful information from the data. Here are some common approaches used in data analysis for medical insurance cost prediction:

1. Descriptive analysis: Descriptive analysis involves summarizing and describing the main characteristics of the data. This may include calculating measures of central tendency (e.g., mean, median, mode), measures of dispersion (e.g., standard deviation, range), and other summary statistics. Descriptive analysis helps in understanding the basic characteristics of the data, identifying trends, and gaining insights into the distribution and variability of the data.
2. Exploratory data analysis (EDA): EDA involves visualizing and exploring the data to identify patterns, trends, and relationships. This may involve creating visualizations such as scatter plots, bar charts, histograms, and box plots to understand the relationships between different variables, detect outliers, and identify potential patterns or trends in the data. EDA helps in gaining a deeper understanding of the data and generating hypotheses that can be tested in further analysis.
3. Correlation analysis: Correlation analysis involves examining the relationships between variables to determine the strength and direction of their association. This may involve calculating correlation coefficients such as Pearson's correlation coefficient or Spearman's rank correlation coefficient to measure the linear or rank relationship between variables. Correlation analysis helps in identifying variables that may be strongly correlated with medical insurance costs and can serve as potential predictors in the predictive models.
4. Predictive modeling: Predictive modeling involves developing statistical or machine learning models to estimate medical insurance costs based on the available data. This may include techniques such as linear regression, decision trees, random forests, support vector machines, or neural networks, depending on the characteristics of the data and the desired accuracy of the predictions. Predictive modeling uses the patterns and relationships uncovered in the data to develop models that can make accurate predictions of medical insurance costs for new cases.
5. Model evaluation: Once the predictive models are developed, they need to be evaluated to assess their accuracy and reliability. This may involve using evaluation metrics such as mean squared error (MSE), root mean

squared error (RMSE), R-squared, or area under the receiver operating characteristic (ROC) curve, depending on the type of model and the prediction task. Model evaluation helps in determining the performance of the models and identifying potential areas for improvement.

6. Interpretation of results: The results of the data analysis and predictive modeling need to be interpreted to extract meaningful insights. This may involve interpreting the coefficients or feature importances in the models, understanding the direction and strength of the relationships between variables, and identifying the key factors that influence medical insurance costs. Interpretation of results helps in understanding the underlying drivers of medical insurance costs and provides insights for decision-making and policy formulation.

Data analysis is a crucial step in the process of medical insurance cost prediction, as it helps in extracting insights from the data, developing predictive models, and evaluating their accuracy and reliability. Proper data analysis techniques and interpretation of results are essential for generating meaningful and reliable predictions of medical insurance costs.

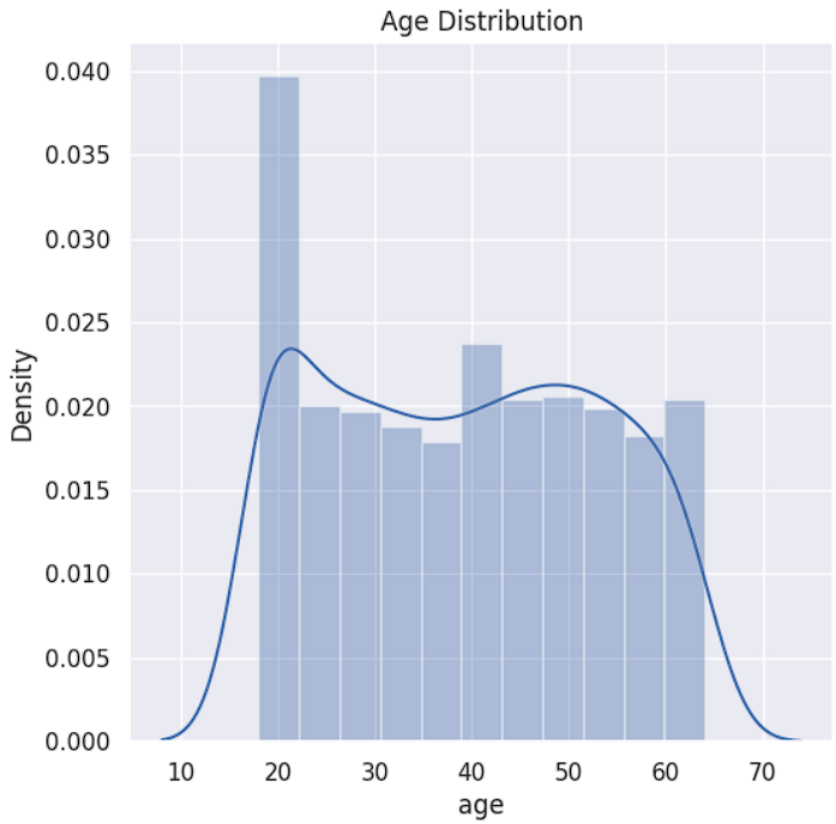
Data Analysis

```
# statistical Measures of the dataset
insurance_dataset.describe()
```

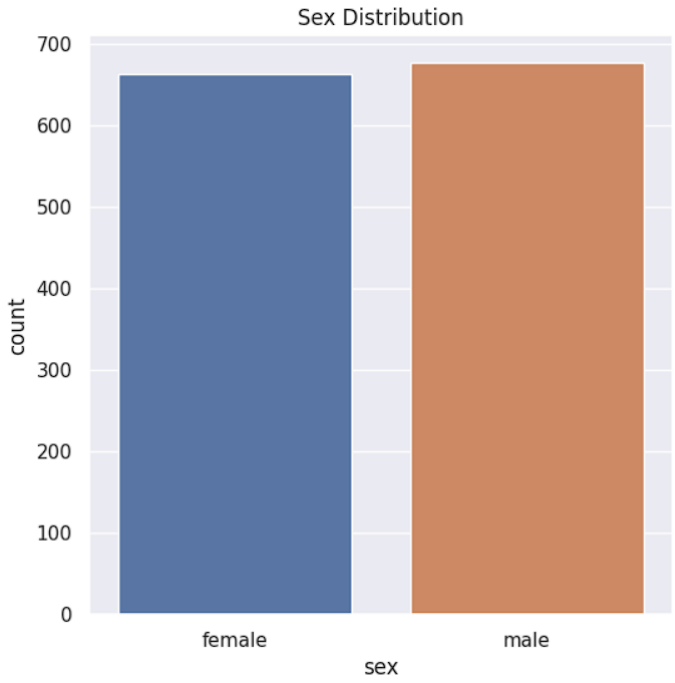
	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	13270.422265
std	14.049960	6.098187	1.205493	12110.011237
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.296250	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	9382.033000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

```
# distribution of age value
sns.set()
plt.figure(figsize=(6,6))
sns.distplot(insurance_dataset['age'])
plt.title('Age Distribution')
plt.show()
```

```
<ipython-input-9-28228e9c3528>: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 `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(insurance_dataset['age'])
```



```
[ ]  # Gender column
     plt.figure(figsize=(6,6))
     sns.countplot(x='sex', data=insurance_dataset)
     plt.title('Sex Distribution')
     plt.show()
```

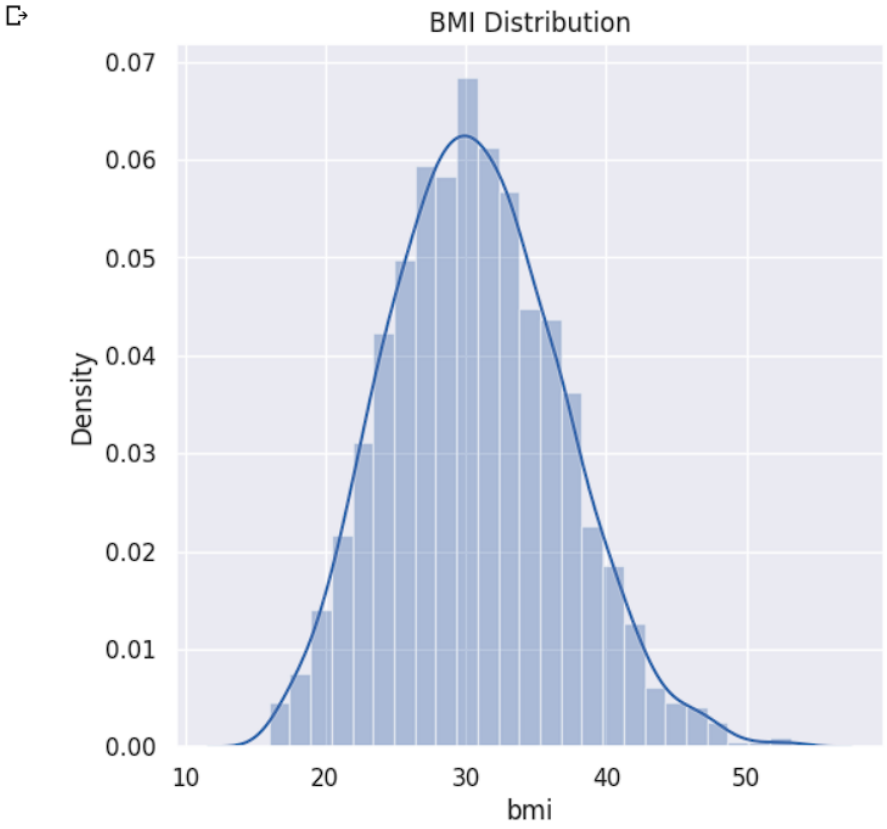



```
[ ]  insurance_dataset['sex'].value_counts()
```

```
male      676
female    662
Name: sex, dtype: int64
```

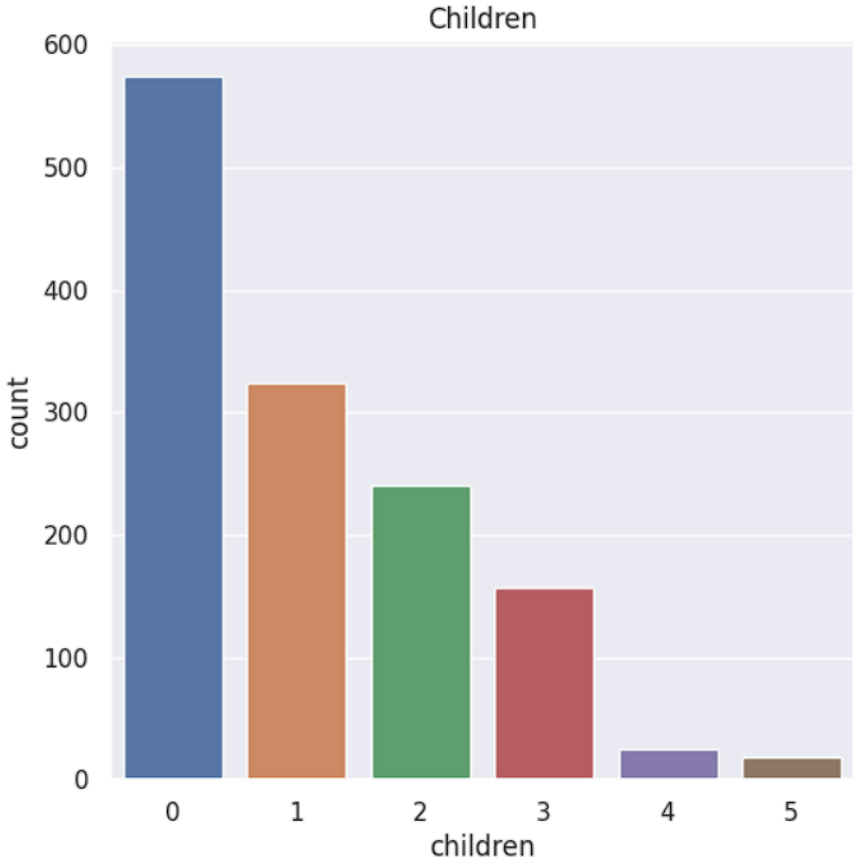
```
[ ]  # bmi distribution
     plt.figure(figsize=(6,6))
     sns.distplot(insurance_dataset['bmi'])
     plt.title('BMI Distribution')
     plt.show()
```

```
<ipython-input-12-81b69896b0d5>: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(insurance_dataset['bmi'])
```



Normal BMI Range --> 18.5 to 24.9

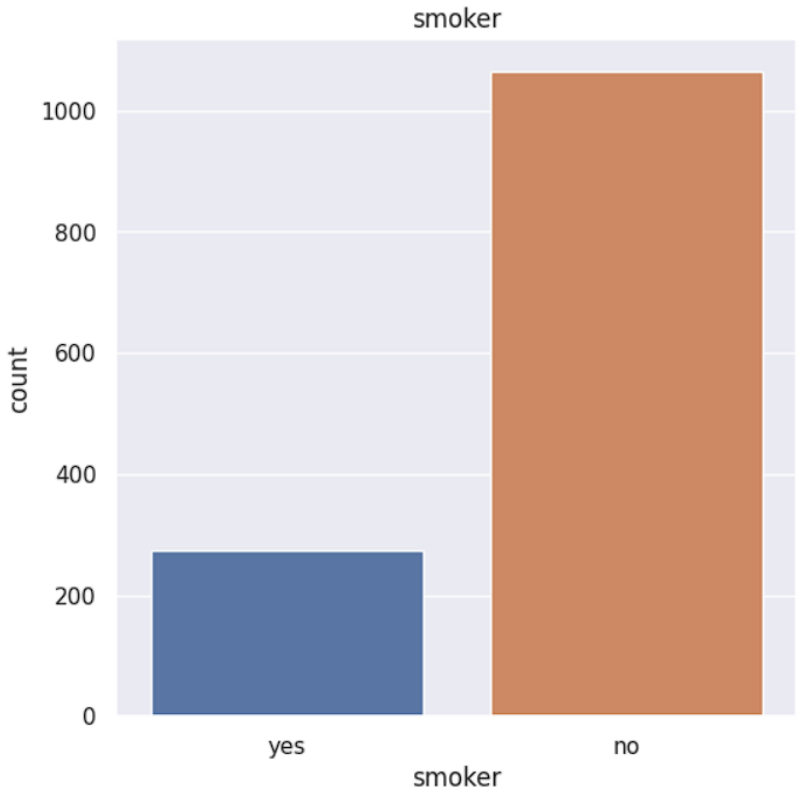
```
# children column
plt.figure(figsize=(6,6))
sns.countplot(x='children', data=insurance_dataset)
plt.title('Children')
plt.show()
```




```
[ ]  insurance_dataset['children'].value_counts()
```

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

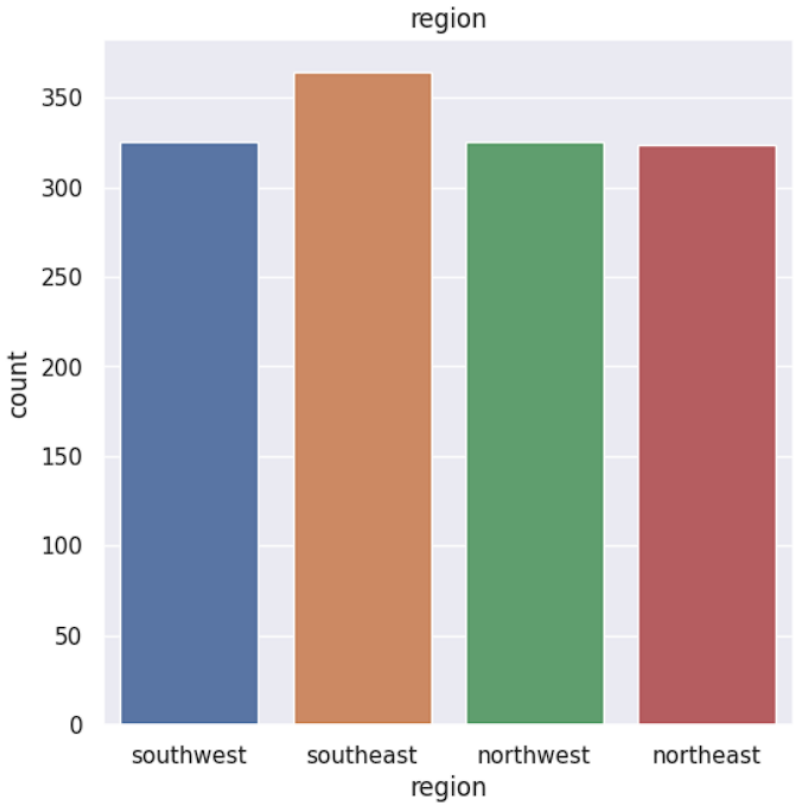
```
[ ]  # smoker column
     plt.figure(figsize=(6,6))
     sns.countplot(x='smoker', data=insurance_dataset)
     plt.title('smoker')
     plt.show()
```




```
[ ]  insurance_dataset['smoker'].value_counts()
```

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

```
[ ]  # region column
     plt.figure(figsize=(6,6))
     sns.countplot(x='region', data=insurance_dataset)
     plt.title('region')
     plt.show()
```

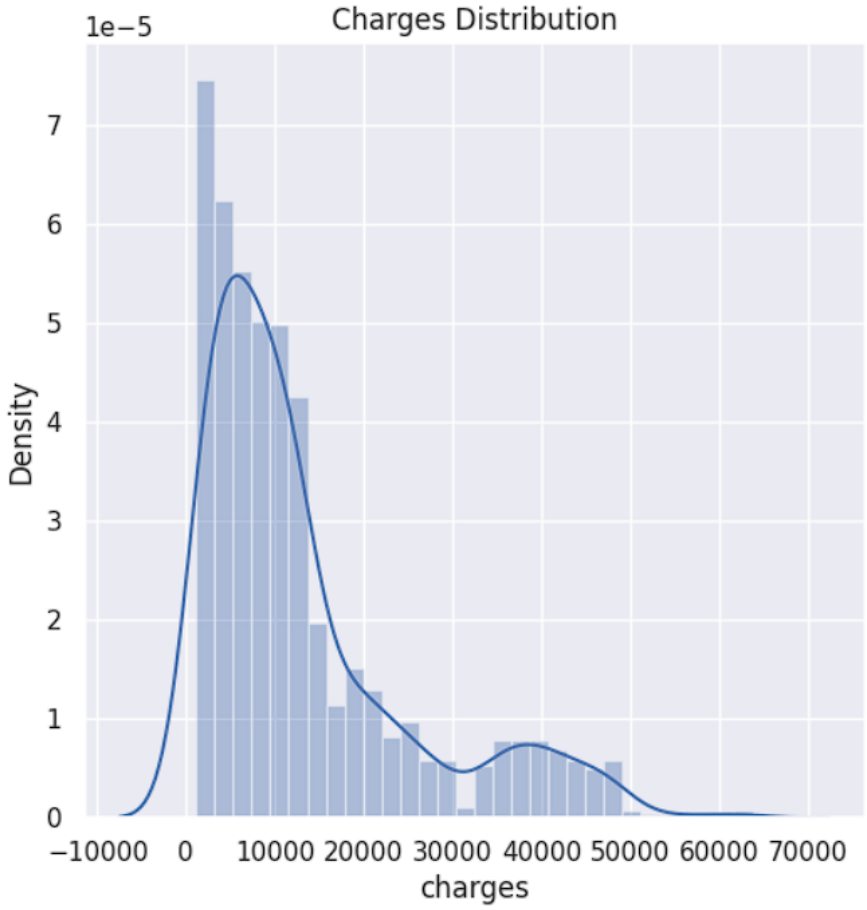

region
350
300
250
200
150
100
50
0
count
southwest
southeast
northwest
northeast
region

```
insurance_dataset['region'].value_counts()
```

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

```
# distribution of charges value
plt.figure(figsize=(6,6))
sns.distplot(insurance_dataset['charges'])
plt.title('Charges Distribution')
plt.show()
```

```
<ipython-input-19-a2fe9b394a51>: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(insurance_dataset['charges'])
```



DATA PREPROCESSING :

Data preprocessing is an important step in the process of medical insurance cost prediction, as it involves cleaning, transforming, and preparing the raw data for analysis. Data preprocessing is essential to ensure that the data is accurate, consistent, and in a format that can be easily analyzed by the predictive models. Here are some common techniques used in data preprocessing for medical insurance cost prediction:

1. Data cleaning: Data cleaning involves identifying and correcting errors, inconsistencies, and inaccuracies in the data. This may include handling missing values, correcting data entry errors, reconciling discrepancies, and removing duplicate records. Data cleaning helps in ensuring that the data used for analysis is accurate and reliable, and reduces the risk of making incorrect predictions due to data errors.
2. Data transformation: Data transformation involves converting the data into a suitable format for analysis. This may include scaling numerical variables to a common range, converting categorical variables into numerical representations (e.g., one-hot encoding), and normalizing or standardizing variables to remove biases. Data transformation helps in preparing the data for analysis and ensuring that all variables are in a consistent format that can be processed by the predictive models.
3. Feature engineering: Feature engineering involves creating new features or variables from the existing data that can potentially improve the predictive accuracy of the models. This may include aggregating variables, creating interaction terms, or extracting relevant information from text or categorical variables. Feature engineering helps in capturing additional information from the data that may not be apparent in the raw data, and can enhance the predictive power of the models.
4. Data integration: Data integration involves combining data from multiple sources, such as different datasets or databases, to create a unified dataset for analysis. This may involve merging datasets based on common variables, handling differences in data formats, and resolving data conflicts. Data integration helps in incorporating diverse sources of data, which can provide a more comprehensive and holistic view of the problem at hand.
5. Data reduction: Data reduction involves reducing the dimensionality of the data by selecting a subset of relevant variables or reducing the number of records in the dataset. This may be done through techniques such as feature selection, where only the most important variables are retained, or data sampling, where a subset of records is used for analysis. Data reduction helps in reducing the computational complexity of the

analysis and can improve the efficiency and performance of the predictive models.

6. Handling imbalanced data: Imbalanced data, where the distribution of classes or outcomes is uneven, can pose challenges in predictive modeling. Data preprocessing techniques such as oversampling minority class, undersampling majority class, or using synthetic data generation methods like SMOTE (Synthetic Minority Over-sampling Technique) can be used to address the issue of imbalanced data and ensure that the models are trained on a balanced dataset.

Data preprocessing is a critical step in the data analysis process as it ensures that the data used for analysis is accurate, consistent, and in a format that can be easily analyzed by predictive models. Proper data preprocessing techniques are essential for generating reliable and accurate predictions of medical insurance costs.

Data Pre-Processing

Encoding the categorical features

```
[ ]  # encoding sex column
     insurance_dataset.replace({'sex':{'male':0,'female':1}}, inplace=True)

     3 # encoding 'smoker' column
     insurance_dataset.replace({'smoker':{'yes':0,'no':1}}, inplace=True)

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

Splitting the Features and Target

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

```
[ ]  print(X)
```

```
      age  sex     bmi  children  smoker  region
0      19    1  27.900         0       0       1
1      18    0  33.770         1       1       0
2      28    0  33.000         3       1       0
3      33    0  22.705         0       1       3
4      32    0  28.880         0       1       3
...   ...  ...     ...       ...     ...     ...
1333   50    0  30.970         3       1       3
1334   18    1  31.920         0       1       2
1335   18    1  36.850         0       1       0
1336   21    1  25.800         0       1       1
1337   61    1  29.070         0       0       3

[1338 rows x 6 columns]
```

```
[ ]  print(Y)
```

```
0       16884.92400
1        1725.55230
2        4449.46200
3       21984.47061
4        3866.85520
            ...
1333    10600.54830
1334     2205.98080
1335     1629.83350
1336     2007.94500
1337    29141.36030
Name: charges, Length: 1338, dtype: float64
```

SPLITTING THE DATA INTO TRAINING DATA & TESTING DATA :

After data preprocessing, the next step in the medical insurance cost prediction process is to split the data into training data and testing data. This is typically done to evaluate the performance and generalization of the predictive models. Here's how it's typically done:

1. Train-Test Split: The dataset is divided into two separate sets: the training data and the testing data. The training data is used to train the predictive models, while the testing data is used to evaluate the performance of the trained models. The ratio of the split depends on the size of the dataset, but a common approach is to use a 70-30 or an 80-20 split, where 70% or 80% of the data is used for training, and the remaining 30% or 20% is used for testing.

2. Random Sampling: The data is randomly sampled to ensure that the training data and testing data are representative of the overall dataset. This helps in avoiding any bias that may arise from using a specific subset of data for training or testing.
3. Stratified Sampling: In case of imbalanced data, where the distribution of classes or outcomes is uneven, stratified sampling may be used to ensure that the representation of different classes or outcomes is maintained in both the training and testing data. This helps in obtaining a balanced distribution of data for training and testing, and ensures that the models are evaluated on all classes or outcomes.
4. Time-based Split: In some cases, if the data has a temporal component, such as time-series data, a time-based split may be used. In this approach, the data is split based on a specific time point, such as using the earlier data for training and the later data for testing. This helps in evaluating the model's ability to generalize to future data.
5. Cross-Validation: Another approach to split the data is to use cross-validation techniques, such as k-fold cross-validation. In this approach, the data is divided into k equally sized folds, and the models are trained and evaluated k times, each time using a different fold as the testing data and the remaining k-1 folds as the training data. This helps in obtaining a more robust estimate of the model's performance, as it evaluates the model on different subsets of the data.

Splitting the data into training and testing data is important to assess the performance and generalization of the predictive models. The models are trained on the training data, and their performance is evaluated on the testing data to ensure that they can make accurate predictions on unseen data. This helps in selecting the best-performing model and provides an estimate of its performance in real-world scenarios

Splitting the data into Training data & Testing Data

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

```
[ ]  print(X.shape, X_train.shape, X_test.shape)

     (1338, 6) (1070, 6) (268, 6)
```

MODEL TRAINING :

After splitting the data into training and testing data, the next step in the medical insurance cost prediction process is to train the predictive models using the training data. Here's an overview of the typical process for model training:

1. Model Selection: Choose the appropriate predictive model(s) for the task at hand. There are various machine learning algorithms that can be used for medical insurance cost prediction, such as linear regression, decision trees, random forests, support vector machines, neural networks, etc. The choice of model(s) depends on the specific requirements of the problem, the size and complexity of the dataset, and the performance metrics of interest.
2. Feature Selection/Engineering: Select the relevant features (i.e., variables or attributes) from the training data that will be used as input to the predictive models. This may involve domain knowledge and exploratory data analysis to identify the most important features that are likely to have an impact on the medical insurance cost. Feature engineering may also involve transforming or encoding categorical variables, handling missing values, or normalizing/standardizing numerical variables to ensure consistent scaling.
3. Model Training: Fit the selected predictive model(s) to the training data. This involves using the training data to learn the patterns and relationships between the input features (independent variables) and the target variable (medical insurance cost or claim amount). The model is trained using various optimization techniques, such as gradient descent, to minimize the prediction error and maximize the model's performance on the training data.
4. Model Evaluation: Assess the performance of the trained models on the testing data. Use the testing data that was set aside earlier to evaluate the performance of the trained models. Common performance metrics used for medical insurance cost prediction include mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), R-squared, and others. These metrics provide insights into how well the models are generalizing to unseen data and making accurate predictions.
5. Hyperparameter Tuning: Optimize the hyperparameters of the predictive models. Hyperparameters are parameters that are not learned during training, but are set by the user prior to model training. Examples of hyperparameters include learning rate, regularization strength, maximum depth of decision trees, etc. Hyperparameter tuning involves

searching different values of hyperparameters and selecting the ones that result in the best performance of the models on the testing data.

6. Model Selection and Deployment: Select the best-performing model based on the evaluation metrics and deploy it for practical use. Once the models are trained, evaluated, and optimized, the best-performing model is selected for deployment in real-world scenarios. The selected model is then used to make predictions on new, unseen data for medical insurance cost prediction.

It's important to note that model training is an iterative process, and multiple iterations may be needed to fine-tune the models and achieve the desired level of accuracy and performance. The training process may also involve trying different algorithms, feature engineering techniques, and hyperparameter tuning to find the optimal combination for the specific problem of medical insurance cost prediction.

LINEAR REGRESSION :

Linear regression is a commonly used statistical method for modeling the relationship between a dependent variable and one or more independent variables. In the context of medical insurance cost prediction, linear regression can be used to build a predictive model that estimates the medical insurance cost based on one or more input features.

Here's an overview of the steps involved in using linear regression for medical insurance cost prediction:

1. Data Preparation: Collect and preprocess the data, as discussed earlier. This may involve gathering data on relevant variables such as age, gender, BMI, smoking status, region, etc., and preparing the data by cleaning, transforming, encoding, and normalizing/standardizing the variables as needed.
2. Feature Selection: Identify the input features (independent variables) that are likely to have an impact on the medical insurance cost. This can be done through domain knowledge, exploratory data analysis, and statistical techniques such as correlation analysis or feature importance analysis.

3. Data Split: Split the data into training and testing datasets. The training dataset will be used to train the linear regression model, while the testing dataset will be used for model evaluation.
4. Model Training: Fit the linear regression model to the training data. This involves estimating the coefficients (slopes) of the linear equation that best fits the training data. The model is trained using the training data to learn the relationship between the input features and the target variable (medical insurance cost).
5. Model Evaluation: Assess the performance of the trained linear regression model on the testing data. Use performance metrics such as mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), R-squared, etc., to evaluate how well the model is generalizing to unseen data and making accurate predictions.
6. Model Optimization: Optimize the model if needed. This may involve fine-tuning the model by adjusting hyperparameters such as learning rate, regularization strength, or model complexity to improve its performance.
7. Model Deployment: Once the linear regression model is trained, evaluated, and optimized, it can be deployed for practical use. The model can be used to make predictions on new, unseen data for medical insurance cost estimation.

Linear regression is a simple and interpretable method for medical insurance cost prediction, but it assumes a linear relationship between the input features and the target variable. It may not be suitable for complex nonlinear relationships in the data. Other machine learning algorithms, such as decision trees, random forests, or neural networks, may be more appropriate in such cases. It's important to select the right model based on the specific requirements of the problem and the characteristics of the data.

Linear Regression

```
[ ]  # loading the Linear Regression model
     regressor = LinearRegression()
```

```
     regressor.fit(X_train, Y_train)
```

```
     LinearRegression
     LinearRegression()
```

```
[ ]  # prediction on training data
     training_data_prediction =regressor.predict(X_train)
```

```
[ ]  # R squared value
     r2_train = metrics.r2_score(Y_train, training_data_prediction)
     print('R squared vale : ', r2_train)
```

```
     R squared vale :  0.751505643411174
```

```
[ ]  # prediction on test data
     test_data_prediction =regressor.predict(X_test)
```

```
[ ]  # R squared value
     r2_test = metrics.r2_score(Y_test, test_data_prediction)
     print('R squared vale : ', r2_test)
```

```
     R squared vale :  0.7447273869684076
```

BUILDING A PREDICTIVE SYSTEM :

Building a predictive system involves several key steps, including:

1. Define the Problem: Clearly define the problem you want to solve with the predictive system. This could be medical insurance cost prediction, as mentioned earlier, or any other specific prediction task, such as disease diagnosis, patient outcome prediction, or medication response prediction, among others.
2. Data Collection and Preprocessing: Collect and preprocess the data needed for the predictive system. This may involve gathering relevant data from various sources, cleaning and transforming the data, handling missing values, encoding categorical variables, and normalizing/standardizing the data as needed.

3. Feature Engineering: Select and engineer appropriate features (input variables) that are likely to have an impact on the prediction task. This may involve domain knowledge, exploratory data analysis, or feature selection techniques to identify the most relevant features for the predictive model.
4. Model Selection: Choose an appropriate machine learning algorithm or predictive modeling technique based on the characteristics of the data and the problem you are trying to solve. There are various options, such as linear regression, decision trees, random forests, support vector machines, neural networks, or ensemble methods, among others. Select the model that best fits your specific problem and data.
5. Data Split: Split the data into training and testing datasets. The training dataset will be used to train the predictive model, while the testing dataset will be used for model evaluation.
6. Model Training: Train the selected predictive model using the training data. This involves feeding the training data into the model, adjusting the model's parameters, and iteratively refining the model to optimize its performance on the training data.
7. Model Evaluation: Evaluate the performance of the trained predictive model on the testing data. Use appropriate performance metrics, such as accuracy, precision, recall, F1 score, or area under the receiver operating characteristic (ROC) curve, to assess how well the model is generalizing to unseen data and making accurate predictions.
8. Model Optimization: Optimize the predictive model if needed. This may involve fine-tuning the model's hyperparameters, such as learning rate, regularization strength, or model complexity, to improve its performance on the testing data.
9. Model Deployment: Once the predictive model is trained, evaluated, and optimized, it can be deployed for practical use. This may involve integrating the model into a larger system or application, creating an API for real-time predictions, or deploying the model on a cloud server for online access.
10. Monitoring and Maintenance: Continuously monitor the performance of the predictive system and update the model as needed. Keep track of new data and changes in the problem domain to ensure the model remains accurate and relevant over time.

Building a predictive system requires careful consideration of data, modeling techniques, performance evaluation, and deployment strategies. It's important to thoroughly understand the problem, select appropriate methods, and validate the model's performance before deploying it in a real-world setting.

Building a Predictive System

```
[ ] input_data = (31,1,25.74,0,1,0)

    # changing input_data to a numpy array
    input_data_as_numpy_array = np.asarray(input_data)

    # reshape the array
    input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

    prediction = regressor.predict(input_data_reshaped)
    print(prediction)

    print('The insurance cost is USD ', prediction[0])
```

```
[3760.0805765]
The insurance cost is USD  3760.080576496057
/usr/local/lib/python3.9/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

CONVERTING THE MONEY TO ONE CURRENCY TO ANOTHER :

Here we are converting the concluded amount into Indian currency because here the concluded currency was in USD so we are converting the USD currency to INR so we need to change it into the Indian currency

converting us dollers to indian currency

```
[ ]  # Function to convert USD to INR
     def usd_to_inr(usd_amount, conversion_rate):
         inr_amount = usd_amount * conversion_rate
         return inr_amount

     # Function to convert number to words
     def num_to_words(num):
         # Define word mappings for numbers
         word_map = {
             0: 'zero',
             1: 'one',
             2: 'two',
             3: 'three',
             4: 'four',
             5: 'five',
             6: 'six',
             7: 'seven',
             8: 'eight',
             9: 'nine',
             10: 'ten',
             11: 'eleven',
             12: 'twelve',
             13: 'thirteen',
             14: 'fourteen',
             15: 'fifteen',
             16: 'sixteen',
             17: 'seventeen',
             18: 'eighteen',
             19: 'nineteen',
```

```
[ ]          20: 'twenty',
             30: 'thirty',
             40: 'forty',
             50: 'fifty',
             60: 'sixty',
             70: 'seventy',
             80: 'eighty',
             90: 'ninety',
         }

         if num <= 20:
             return word_map[num]
         elif num < 100:
             tens_digit = num // 10 * 10
             ones_digit = num % 10
             return word_map[tens_digit] + '-' + word_map[ones_digit]
         elif num < 1000:
             hundreds_digit = num // 100
             tens_ones_digit = num % 100
             return word_map[hundreds_digit] + ' hundred ' + num_to_words(tens_ones_digit)
         elif num < 100000:
             thousands_digit = num // 1000
             hundreds_tens_ones_digit = num % 1000
             return num_to_words(thousands_digit) + ' thousand ' + num_to_words(hundreds_tens_ones_digit)
         elif num < 10000000:
             lakhs_digit = num // 100000
             thousands_hundreds_tens_ones_digit = num % 100000
             return num_to_words(lakhs_digit) + ' lakh ' + num_to_words(thousands_hundreds_tens_ones_digit)
         else:
             return 'Number out of range'
```

```
[ ]
    # Conversion rate from USD to INR
    conversion_rate = 81.82

    # Input: USD amount
    usd_amount = float(input("Enter USD amount: "))

    # Convert USD to INR
    inr_amount = usd_to_inr(usd_amount, conversion_rate)

    # Convert INR amount to words
    inr_amount_words = num_to_words(int(inr_amount))

    # Display the converted amount in INR in words
    print("USD:", usd_amount)
    print("INR:", inr_amount)
    print("INR in words:", inr_amount_words)
```

```
Enter USD amount: 3760
USD: 3760.0
INR: 307643.19999999995
INR in words: three lakh seven thousand six hundred forty-three
```

S