Aiml Mini Project Report
on
Insurance Cost Predictor

by

Manish Shashikant Jadhav
2023301005

Mayur Krishna Solankar
2023301018

Vishesh Bhimji Savani
2022300100

Guide Name:
Prof. Swapnali Kurhade

Computer Engineering Department

Bharatiya Vidya Bhavan's

Sardar Patel Institute of Technology

Munshi Nagar, Andheri(W), Mumbai-400058

University of Mumbai

Nov 2024

1. Introduction

1.1 Problem Statement

To develop a machine learning-based predictive model for accurately estimating health insurance costs by analyzing key variables such as age, BMI, number of children, gender, smoking status, and region, addressing the challenges of manual cost estimation and improving the efficiency of insurance pricing strategies.

Key Challenges:

- Manual insurance cost calculation is time-consuming
- High potential for human error
- Complex interactions between multiple factors affecting insurance charges
- Need for an automated, data-driven approach to cost prediction

1.2 Literature Survey/Market Survey

Research Paper Link:

https://www.researchgate.net/publication/348559741_Predict_Health_Insurance_Cost_by_using _Machine_Learning_and_DNN_Regression_Models

1.3 Research Objectives

- Analyze factors influencing health insurance costs
- Implement linear regression and random forest models
- Generate predictive model for insurance cost estimation

2.Methodology

3.1 Data Source

- **Dataset:** Kaggle Medical Cost Personal Dataset
- **Features:** Age, BMI, number of children, gender, smoking status, region
- **Target Variable:** Medical insurance charges

3.2 Data Preprocessing

- Cleaned and prepared dataset
- Converted categorical variables to numeric values
- Split dataset into training and testing sets
- Created pickle file for random forest model

3.3 Machine Learning Models

1. Linear Regression
2. Random Forest Regression

3.4 Technology Stack

- Backend: Django (Python)
- Frontend: React.js
- Machine Learning: scikit-learn, pickle

3.5 System Architecture

3.5.1 Backend Development (Django)

- Create Django REST Framework for API endpoints
- Implement machine learning model loading from pickle file
- Develop prediction logic for insurance cost estimation
- Handle data validation and preprocessing
- Create secure API for model inference

3.5.2 Frontend Development (React)

- Design responsive user interface
- Create input forms for insurance variables
- Implement state management
- Develop prediction result display
- Connect frontend with Django backend API

3.5.3 Machine Learning Pipeline

- Data preprocessing
- Linear regression model
- Random forest regression model
- Model serialization using pickle
- Performance evaluation metrics

3.5.4 Integration Approach

- Django backend serves machine learning model
- React frontend consumes prediction API
- Seamless communication between frontend and backend
- Secure data transmission

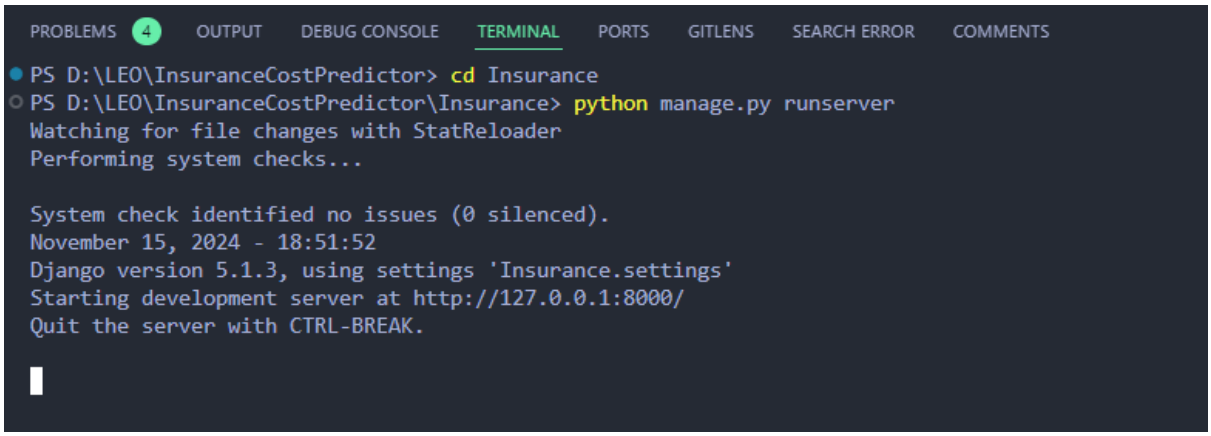
3. Implementation

Github Repo Link:

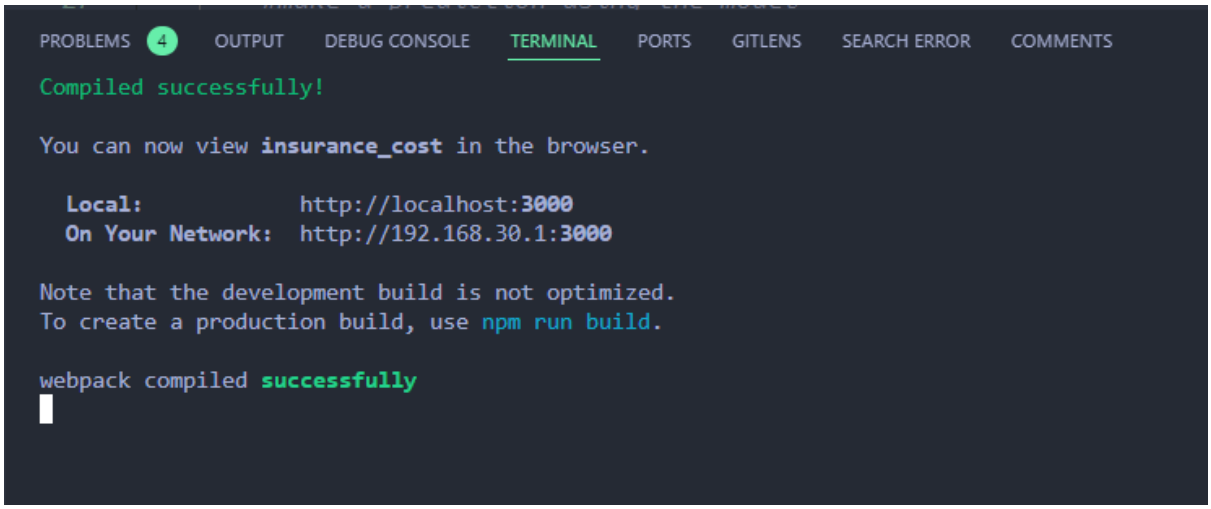
https://github.com/manishjadhav9/InsuranceCostPredictor

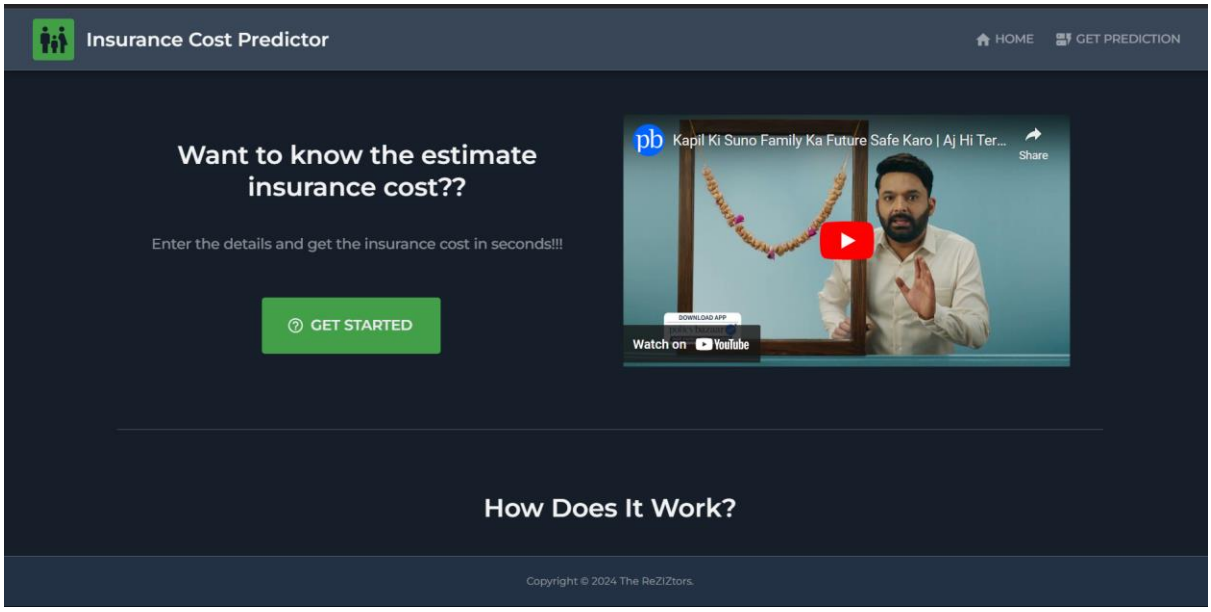
Collab Link:
https://colab.research.google.com/drive/1KF87J81hl5vbbIUWB_9pvp9Vj8L0BMGC

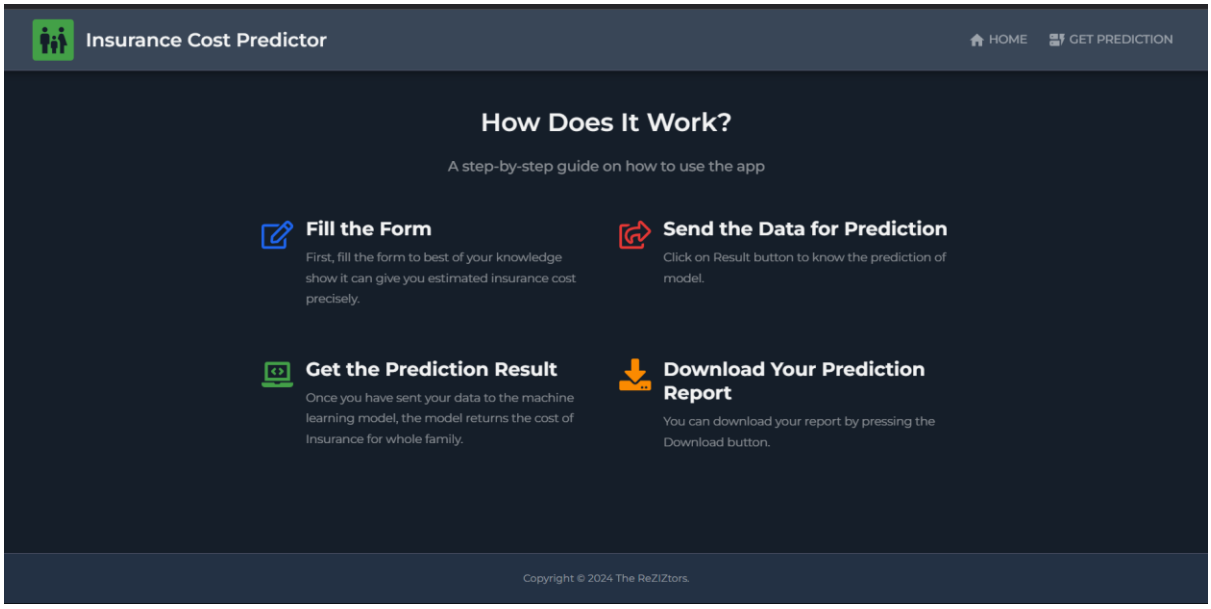
1. **Start Server:**

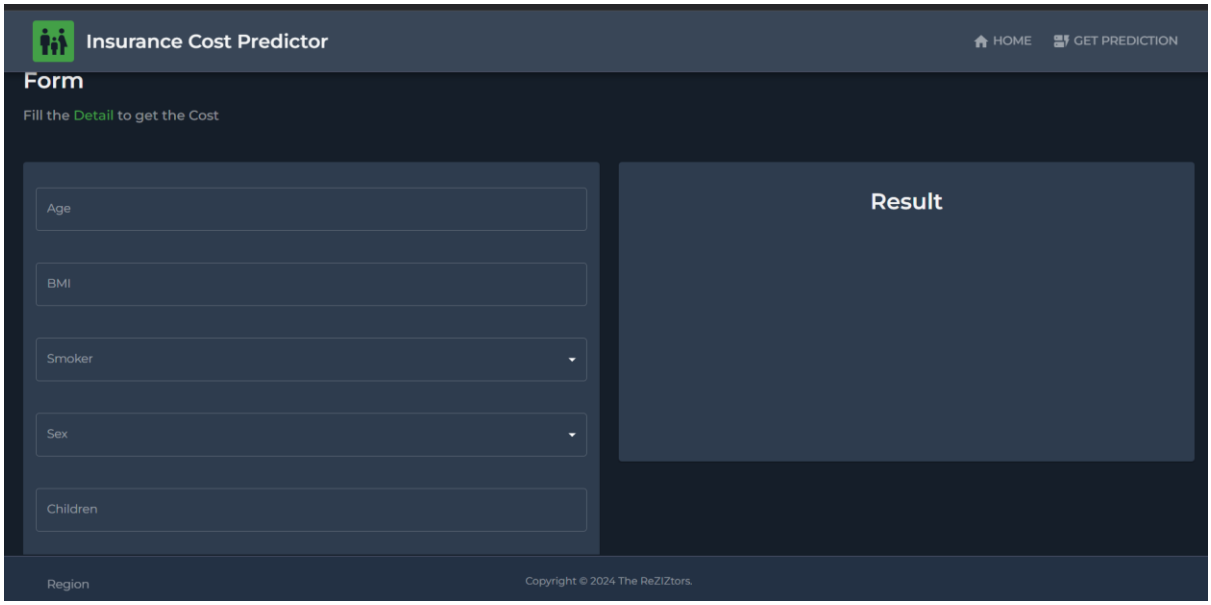


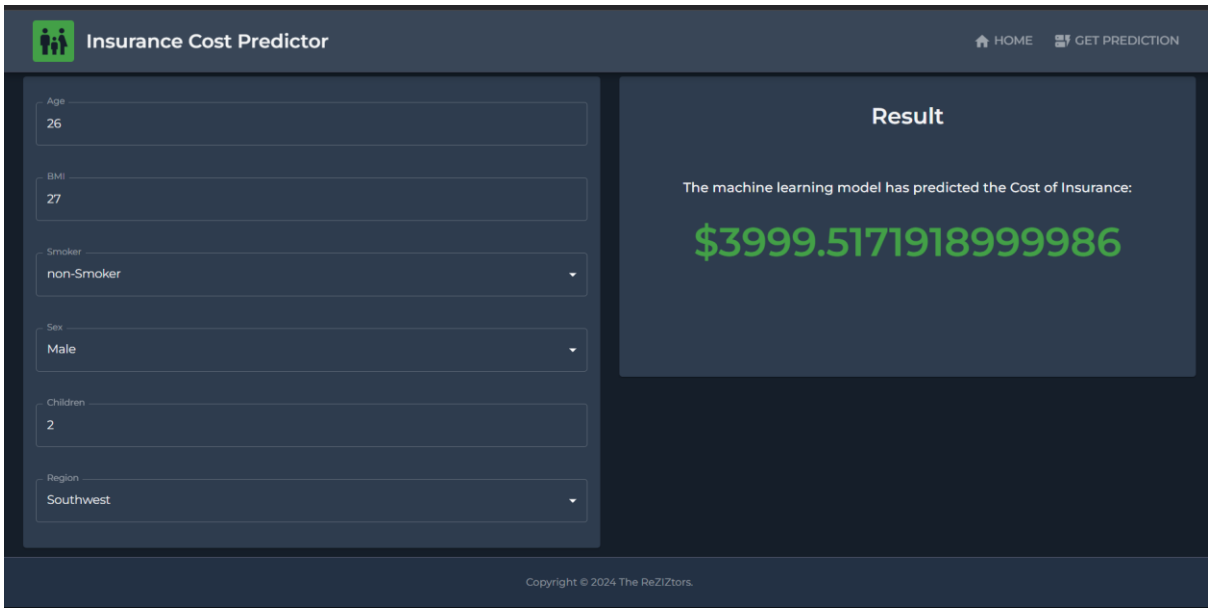
2. **Start frontend:**

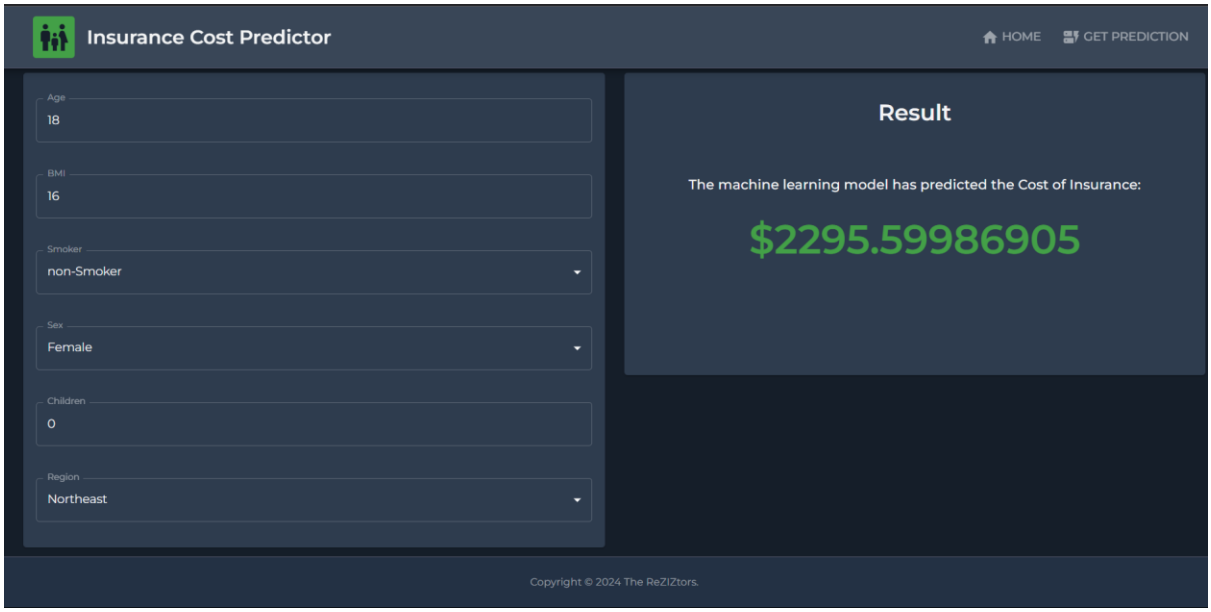



Insurance Cost Predictor
HOME
GET PREDICTION
Want to know the estimate insurance cost??
Enter the details and get the insurance cost in seconds!!!
GET STARTED
pb
Kapil Ki Suno Family Ka Future Safe Karo | Aj Hi Ter...
Share
Watch on YouTube
How Does It Work?
Copyright © 2024 The ReZIZtors.

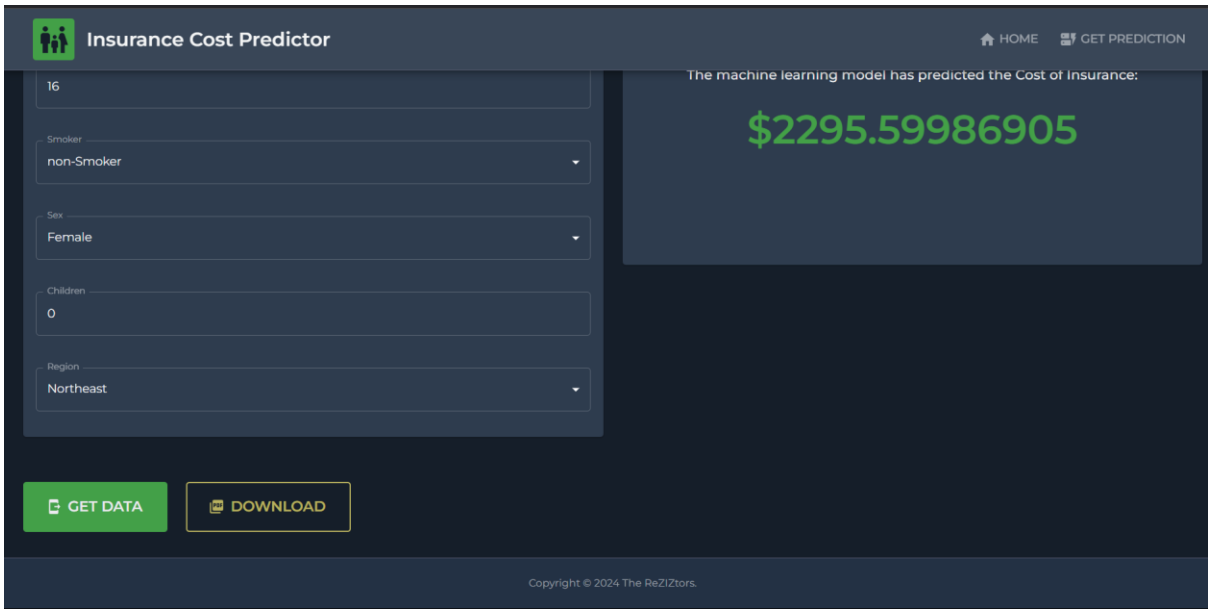

Insurance Cost Predictor
HOME
GET PREDICTION
How Does It Work?
A step-by-step guide on how to use the app
Fill the Form
First, fill the form to best of your knowledge show it can give you estimated insurance cost precisely.
Send the Data for Prediction
Click on Result button to know the prediction of model.
Get the Prediction Result
Once you have sent your data to the machine learning model, the model returns the cost of Insurance for whole family.
Download Your Prediction Report
You can download your report by pressing the Download button.
Copyright © 2024 The ReZIZtors.

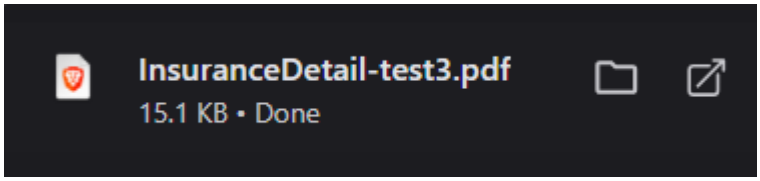

Insurance Cost Predictor
HOME
GET PREDICTION
Form
Fill the Detail to get the Cost
Age
BMI
Smoker
Sex
Children
Region
Result
Copyright © 2024 The ReZIZtors.


Insurance Cost Predictor
HOME
GET PREDICTION
Age
26
BMI
27
Smoker
non-Smoker
Sex
Male
Children
2
Region
Southwest
Result
The machine learning model has predicted the Cost of Insurance:
$3999.5171918999986
Copyright © 2024 The ReZIZtors.



3. Download the result:





4. Conclusion

The **Insurance Cost Predictor Project** is a testament to how technology can streamline financial planning and decision-making in the insurance sector. By leveraging machine learning algorithms, this project enables accurate predictions of insurance costs based on key factors such as age, gender, BMI, smoking habits, and more.

This tool not only benefits customers by offering transparency in cost estimation but also aids insurance companies in optimizing their pricing strategies, enhancing customer satisfaction, and promoting fairness. The project showcases the immense potential of data science in solving real-world problems, emphasizing the importance of predictive analytics in improving efficiency and decision-making across industries.

5. References

- https://www.kaggle.com/datasets/mirichoi0218/insurance?select=insurance.csv
- https://www.researchgate.net/publication/348559741_Predict_Health_Insurance_Cost_by_using _Machine_Learning_and_DNN_Regression_Models

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
```

```
insurance_dataset = pd.read_csv('/content/insurance.csv')
```

```
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

```
insurance_dataset.shape
```

```
(1338, 7)
```

```
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()
```

	0
age	0
sex	0
bmi	0
children	0
smoker	0
region	0
charges	0

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

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)
```

```
<ipython-input-8-7d5826986d65>:2: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed in a future ver
  insurance_dataset.replace({'sex':{'male':0,'female':1}}, inplace=True)
<ipython-input-8-7d5826986d65>:5: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed in a future ver
  insurance_dataset.replace({'smoker':{'yes':0,'no':1}}, inplace=True)
<ipython-input-8-7d5826986d65>:8: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed in a future ver
  insurance_dataset.replace({'region':{'southeast':0,'southwest':1,'northeast':2,'northwest':3}}, inplace=True)
```

```
insurance_dataset.head()
```

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

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

```
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

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

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

```
LinearRegression
LinearRegression()
```

Model Evaluation

```
# 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
```

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

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

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

```
[[31.    1.   25.74  1.    1.    0.  ]]
[4340.35495946]
The insurance cost is USD  4340.354959456534
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:493: UserWarning: X does not have valid feature names, but LinearRegression
  warnings.warn(
```

Training the Random Forest Model

```
# Assuming 'charges' is the target variable and others are features
X = insurance_dataset.drop('charges', axis=1)  # Features (all columns except target)
y = insurance_dataset['charges']  # Target (the 'charges' column)

# Handling categorical variables using one-hot encoding
X = pd.get_dummies(X, drop_first=True)  # Drop the first category to avoid multicollinearity
```

```
# Checking the preprocessed features
X.head()
```

	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

Splitting the data

```
# Splitting the data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Checking the shape of the splits to verify
print(f"Training data shape: X_train = {X_train.shape}, y_train = {y_train.shape}")
print(f"Testing data shape: X_test = {X_test.shape}, y_test = {y_test.shape}")
```

```
Training data shape: X_train = (1070, 6), y_train = (1070,)
Testing data shape: X_test = (268, 6), y_test = (268,)
```

```
from sklearn.ensemble import RandomForestRegressor  # For regression tasks

# Creating a RandomForestRegressor model
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)

# Training the model on the training data
rf_model.fit(X_train, y_train)
```

```
RandomForestRegressor
RandomForestRegressor(random_state=42)
```

Making Predictions

```
# Predicting the target variable on the test data
y_pred = rf_model.predict(X_test)

# Showing the first few predictions to get a sense of the output
y_pred[:10]
```

```
array([ 9964.4411712,  5614.908105 , 28122.751816 , 12317.3170911,
       34592.244544 ,  8330.8161489,  2185.8946415, 14516.191212 ,
        5719.9984878, 10166.264253 ])
```

EValuate the model

```
# Calculating the Mean Squared Error and Root Mean Squared Error
mse = metrics.mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)

# Printing the evaluation metrics
print(f"Mean Squared Error: {mse}")
print(f"Root Mean Squared Error: {rmse}")
```

```
Mean Squared Error: 20605006.150018733
Root Mean Squared Error: 4539.273746979657
```

```
# Getting the feature importances from the trained model
feature_importances = rf_model.feature_importances_

# Creating a DataFrame to display feature names and their importance
importance_df = pd.DataFrame({
    'Feature': X.columns,
    'Importance': feature_importances
})
```

```
# Sorting the features by importance in descending order
importance_df = importance_df.sort_values(by='Importance', ascending=False)

# Displaying the feature importance
print(importance_df)
```

```
      Feature  Importance
4    smoker    0.608618
2       bmi    0.216403
0       age    0.134356
3  children    0.019627
5    region    0.014326
1       sex    0.006670
```

```
# Assuming the trained RandomForest model is stored in 'rf_model'
input_data = (31, 1, 25.74, 1, 1, 0)  # Example input data

# Converting the input_data into a numpy array
input_data_as_numpy_array = np.asarray(input_data)

# Reshaping the array to match the model input format (1 row, multiple columns)
input_data_reshaped = input_data_as_numpy_array.reshape(1, -1)

# Printing the reshaped input data for reference
print(input_data_reshaped)

# Using the trained Random Forest model to make a prediction
prediction = rf_model.predict(input_data_reshaped)

# Printing the predicted insurance cost
print(f'The predicted insurance cost is USD {prediction[0]}')
```

```
[[31.    1.   25.74  1.    1.    0.  ]]
The predicted insurance cost is USD 4878.8846039
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:493: UserWarning: X does not have valid feature names, but RandomForestRegre
  warnings.warn(
```

```
import pickle
import joblib
filename='InsuranceCostPredictor.pkl'
joblib.dump(rf_model, filename)
```

```
['InsuranceCostPredictor.pkl']
```