Name	Manish Shashikant Jadhav, Mayur Krishna Solankar, Vishesh Bhimji Savani
UID no.	2023301005, 2023301018, 2023300100

Experiment 9 & 10	
AIM :	Mini Project
TITLE:	Insurance Score Predictor
Theory:	**The Insurance Score Predictor** project aims to develop a machine learning model that can predict insurance costs based on medical records data. The project consists of the following key components: **1. Data Pre-processing:** The medical records data was preprocessed using Google Colab, which involved cleaning, handling missing values, and preparing the data for model training. **2. Model Training:** Two machine learning models were trained on the preprocessed data: - Linear Regression - Random Forest The best performing model, which was the Random Forest model, was saved as a pickle file for deployment. **3. Web Application:** A web application was developed using Django (backend) and React (frontend) to allow users to interact with the insurance cost prediction model. The saved Random Forest model pickle file was integrated into the Django application. **Data Preprocessing (Google Colab)** The data preprocessing steps performed in Google Colab included: • Importing the necessary libraries (e.g., pandas, numpy, sklearn) • Loading the medical records dataset • Handling missing values

- Encoding categorical variables
- Splitting the data into training and testing sets
- Scaling the features using standard scaler

Model Training

Two machine learning models were trained on the preprocessed data:

1. Linear Regression:

- The linear regression model was trained on the training data.
- The model's performance was evaluated on the test data.
- The model's pickle file was saved for deployment.

2. Random Forest:

- The random forest model was trained on the training data.
- The model's performance was evaluated on the test data.
- The model's pickle file was saved for deployment.

The Random Forest model was found to have better performance and was selected for integration into the web application.

Web Application (Django and React)

The web application was built using Django for the backend and React for the frontend. The key features of the web application include:

1. User Interface: The React frontend provides a clean and intuitive user interface for interacting with the insurance cost prediction model.

2. Model Integration: The saved Random Forest model pickle file was integrated into the Django backend, allowing the web application to utilize the trained machine learning model for predictions.

	3. Prediction Functionality: Users can input their medical records data into the web application, and the application will use the integrated Random Forest model to predict their insurance costs. **4. Responsive Design:** The web application was designed to be responsive, ensuring a seamless user experience across different devices and screen sizes. By combining the data preprocessing, model training, and web application components, the Insurance Score Predictor project provides a comprehensive solution for predicting insurance costs based on medical records data.
Collab Link:	https://colab.research.google.com/drive/1KF87J81hl5vbbIUWB_9pvp9Vj8L0BMGC
Github Repo Link:	https://github.com/manishjadhav9/InsuranceCostPredictor

```
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

```
dtype: int64
```

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      i ?
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]}')
```

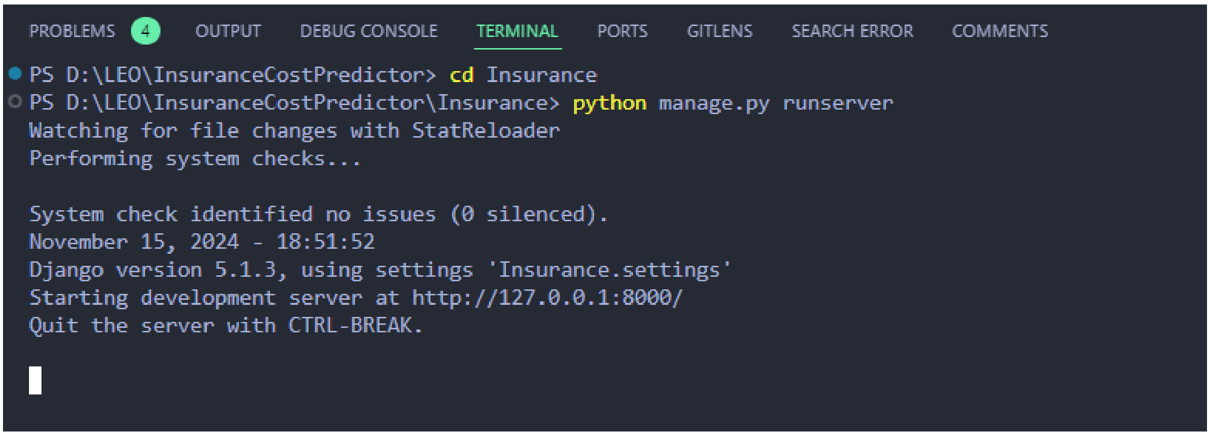
```
[[31.    1.   25.74  1.    1.    0.  ]]
The predicted insurance cost is USD 4878.8846039
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:493: UserWarning: X does not have valid feature names, but RandomForestRegre
  warnings.warn(
```

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

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

Project Implementation:

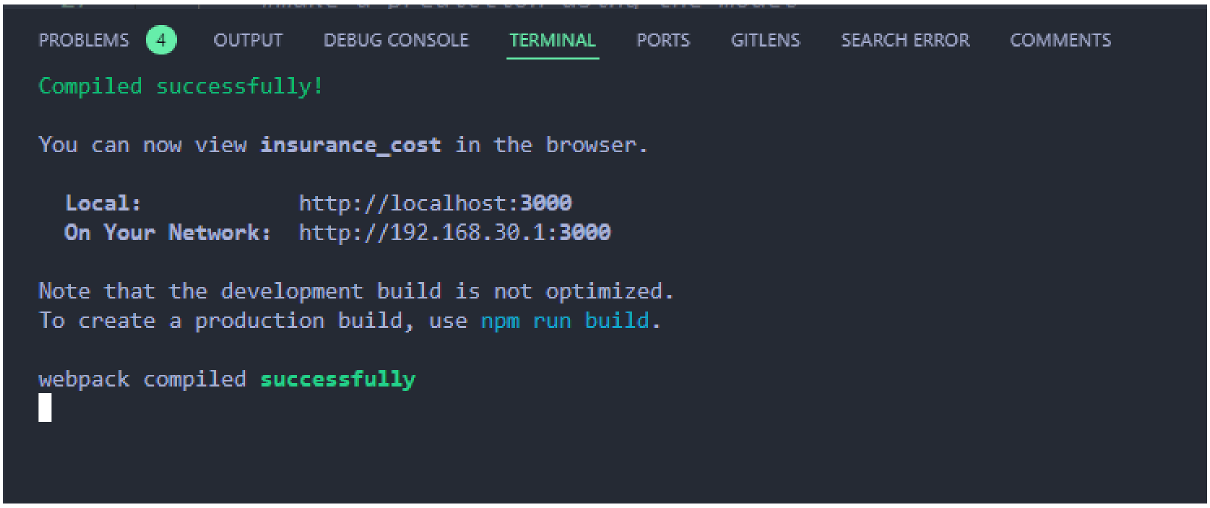
1. Start Server:



```
PS D:\LEO\InsuranceCostPredictor> cd Insurance
PS D:\LEO\InsuranceCostPredictor\Insurance> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
November 15, 2024 - 18:51:52
Django version 5.1.3, using settings 'Insurance.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

2. Start frontend:



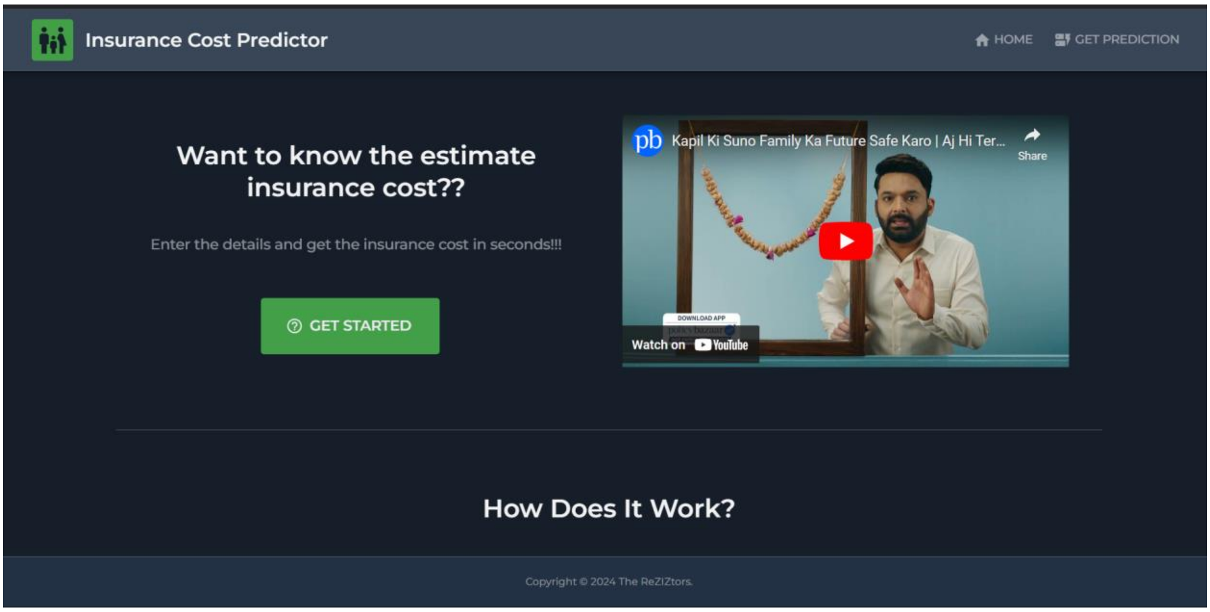
```
Compiled successfully!

You can now view insurance_cost in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.30.1:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

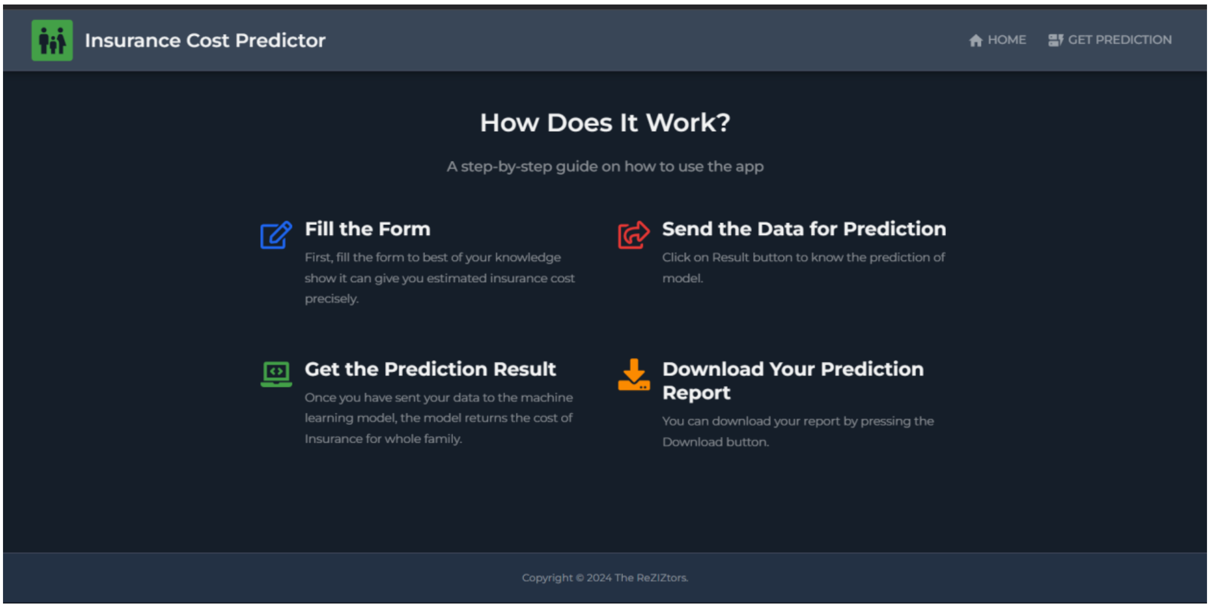


Want to know the estimate insurance cost??

Enter the details and get the insurance cost in seconds!!!

How Does It Work?

Copyright © 2024 The ReZIZtors.



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.

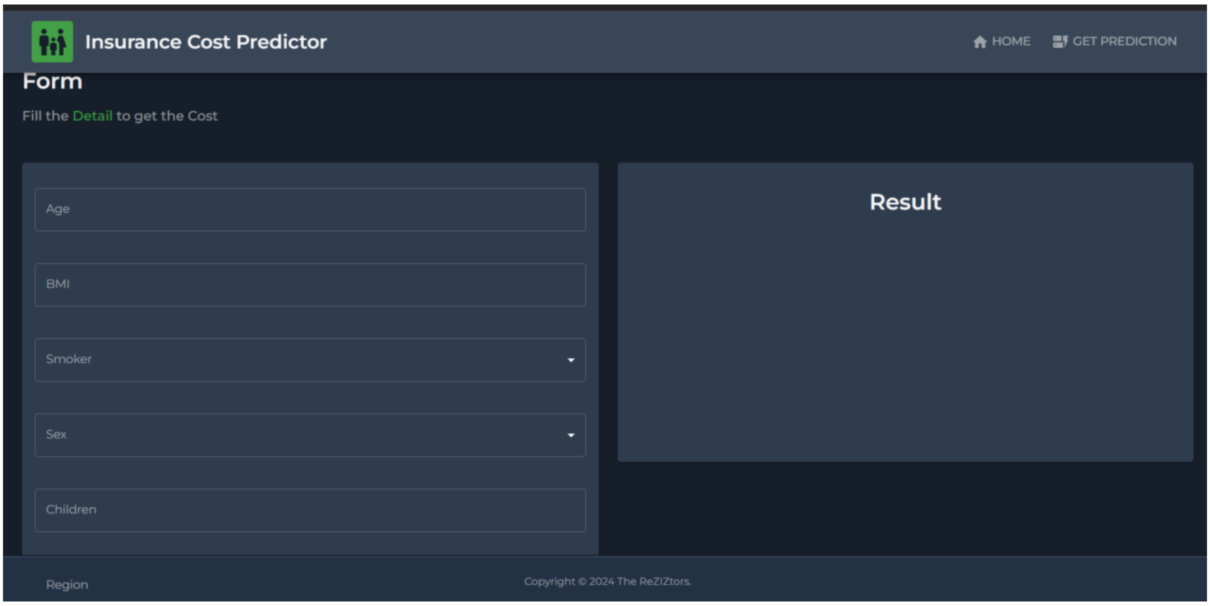
Get the Prediction Result

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.

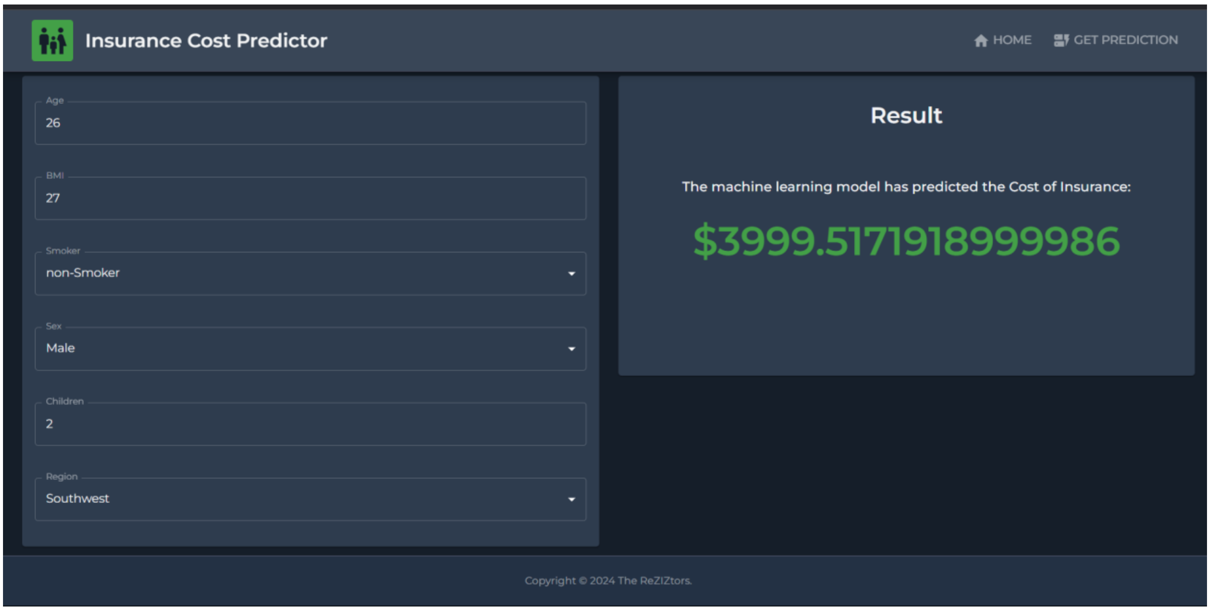


Form

Fill the Detail to get the Cost

Result

Copyright © 2024 The ReZIZtors.



Age: 26

BMI: 27

Smoker: non-Smoker

Sex: Male

Children: 2

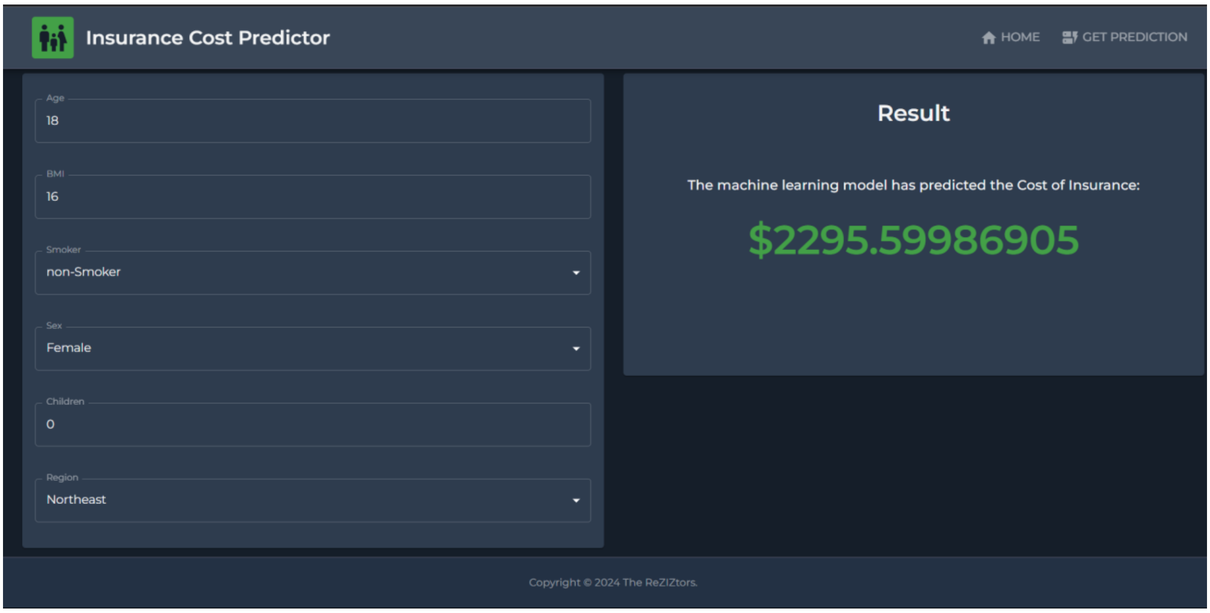
Region: Southwest

Result

The machine learning model has predicted the Cost of Insurance:

$3999.5171918999986

Copyright © 2024 The ReZIZtors.



Age: 18

BMI: 16

Smoker: non-Smoker

Sex: Female

Children: 0

Region: Northeast

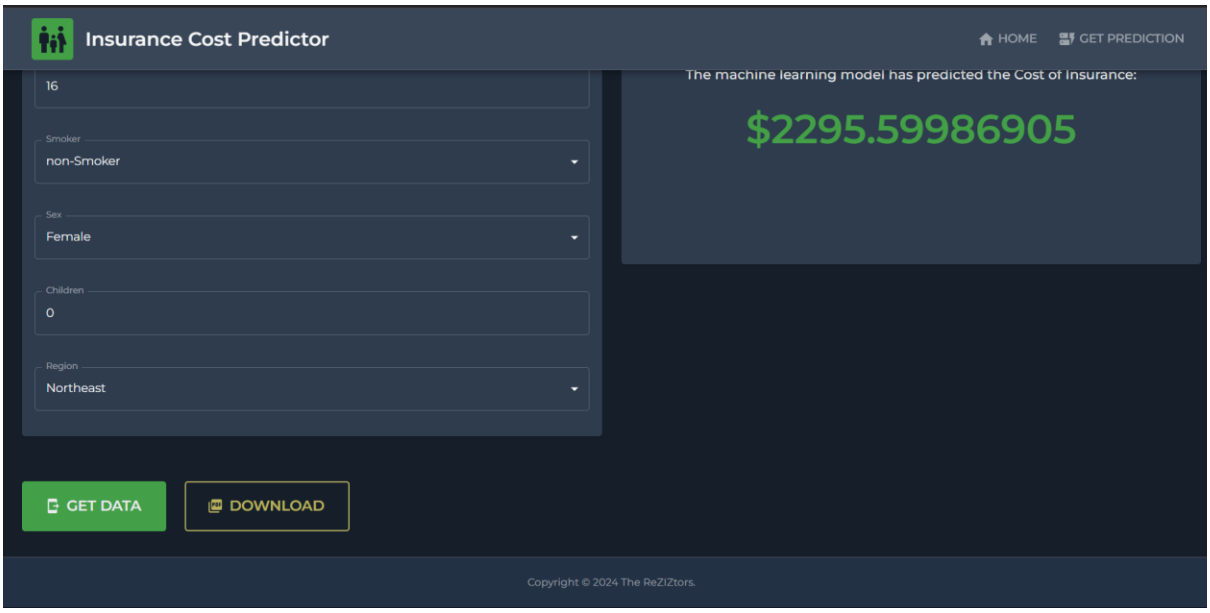
Result

The machine learning model has predicted the Cost of Insurance:

$2295.59986905

Copyright © 2024 The ReZIZtors.

3. **Download the result:**



16

Smoker: non-Smoker

Sex: Female

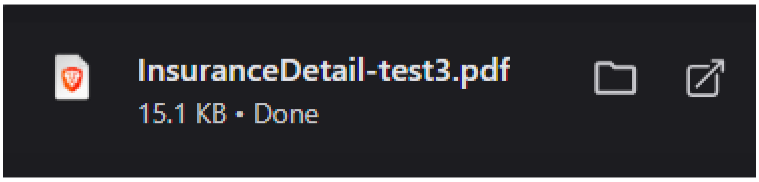
Children: 0

Region: Northeast

The machine learning model has predicted the Cost of Insurance:

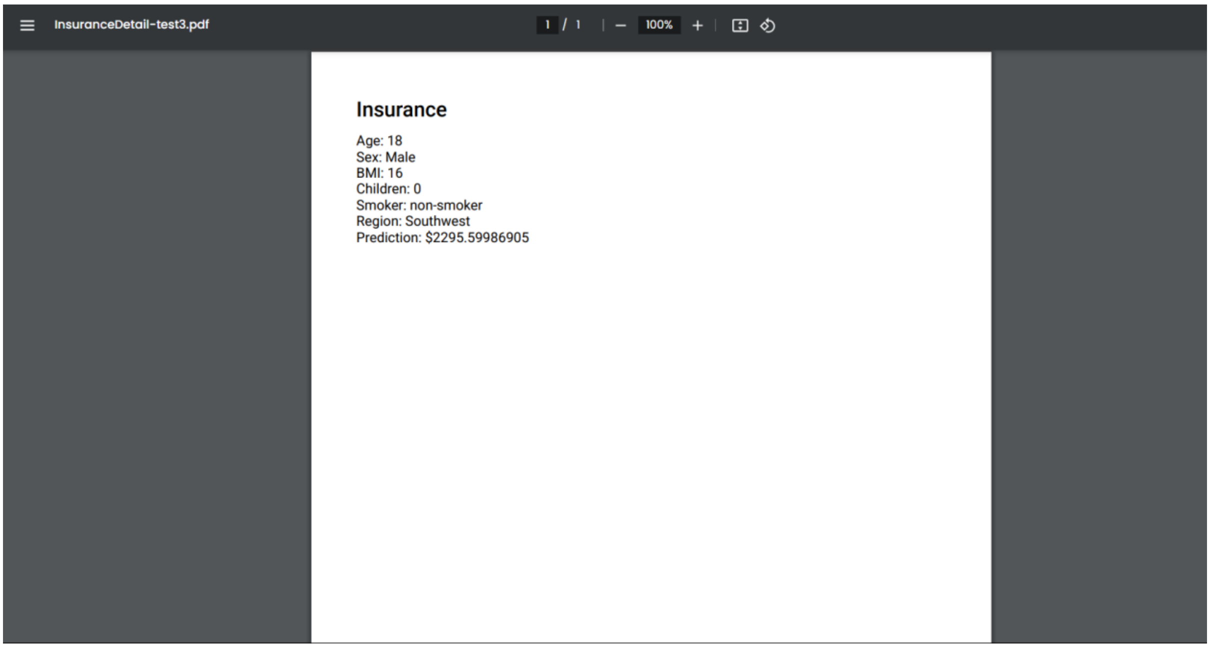
$2295.59986905

Copyright © 2024 The ReZIZtors.



InsuranceDetail-test3.pdf

15.1 KB • Done



Insurance

Age: 18
Sex: Male
BMI: 16
Children: 0
Smoker: non-smoker
Region: Southwest
Prediction: $2295.59986905