20

21

Data Mining

Medical insurance cost prediction

Priyansha Grover -N015
Arjun Khare-N025

Project

Data Mining Analysis and Prediction in Python Using Dataset from Kaggle

20

21

Table of contents

Data Mining

Project

01

INTRODUCTION

This project builds a prediction system from given dataset of Medical Insurance Cost Records, using concepts of Data Mining such as Linear Regression and Train Test Split. The language used for execution is python. Dataset was used for training the models and that training helped to come up with some predictions. Then the predicted amount was compared with the actual data to test and verify the model. Later the accuracies of these models were compared.

PROBLEM STATEMENT

- A Machine Learning system needs to be built for a Medical Insurance Company that can learn from the data and analyse the data to predict what the cost of Medical Insurance will be for certain customers.
- This is required for automatic, accurate and fast prediction from historical data, as calculating it individually for every customer for the company employees will be a tedious task.
- It is a very complex method people can be fooled easily about the amount of the insurance and may unnecessarily buy some expensive health insurance, hence an automatic prediction system is necessary.

SCOPE

Current Scope of this project involves:

- Data collection and analysis
- Data Pre-Processing: Encoding the categorical features, Splitting the Features and Target, Splitting the data into Training data & Testing Data
- Model Training: Linear Regression, Model Evaluation
- Building a Predictive System

SCOPE

Future Scope of this project involves:

- This project is based on a single machine learning model, hence i other models like multiple linear regression, decision tree and gradient boosting algorithms can be included for better results and better accuracy.
- This project uses one particular dataset, hence it needs to be modified for other datasets if need be.

Proposed Architecture

01	02	03	04
Data Analysis	**Data Pre-processing**	**Train Test Split**	**Linear Regression Model**
Insurance Cost Dataset is first acquired and analysed.	Data Encoding is done to convert Categorical Featured to numerical.	We split the data into training data and testing data for comparison of accuracy.	We convert it into a trained Linear Regression Model, We can then input New Data for future prediction

Data Mining

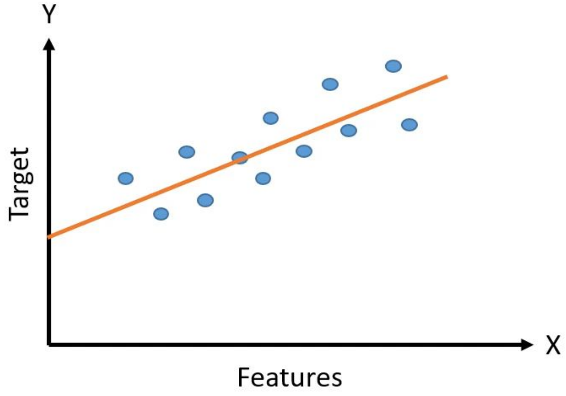
Proposed Algorithms

Linear Regression :

- Linear regression is a basic and commonly used type of predictive analysis.
- It is a linear approach for modelling the relationship between a scalar response and one or more explanatory variables.
- It has two variables used, one that is dependant and the other independent. The independent variables are fitted into the x-axis through which the dependant variables are calculated or predicted on the y-axis, based on any given information or data, or slope and intercept.
- The formula that linear regression follows is : Y = MX + C.
- Y is the dependant variable, X is the independent variable.
- M is the slope of the curve, and C is the constant or intercept.
- In this project, some new data is inputted into the program, which inputs to the X-axis of the graph.
- This value at the X-axis finds an intercept on the curve and finds the resultant Y-axis value.

Project

Linear Regression



X – input features

Y – Prediction Probability

M – Slope

C - Intercept

$$Y = mX + c$$

Line Equation

LITERATURE REVIEW

Designing and Implementation :

Importing Dependancies :

Importing the Dependencies

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
```

DESIGN AND IMPLEMENTATION

- NumPy can be used to perform a wide variety of mathematical operations on arrays. It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices.
- Pandas is just a library of python used for data manipulation and analysis. It also reads csv files as datasets, which makes it easier to do analysis.
- Matplotlib is a cross-platform, data visualization and graphical plotting library for Python.
- Seaborn is an open-source Python library built on top of matplotlib. It is used for data visualization and exploratory data analysis. Seaborn works easily with dataframes and the Pandas library.
- To divide a given dataset into train data and test data, we have imported train_test_split.
- To predict data based on a given input and data we have imported LinearRegression.
- Metrics is imported to evaluate the accuracy of the predictive model used for the given dataset by finding r squared value.

DESIGN AND IMPLEMENTATION

Data Collection & Analysis

- Data is loaded as a dataset from a csv file.
- Number of rows and columns are found out.
- Information regarding the columns of the dataset are analysed.
- The dataset is checked for any null values.
- The dataset is described to see the minimum value, maximum value, standard deviation, and 25%, 50%, 75% quartiles.
- The graphs of all columns(age, sex, bmi, children, smoker, region, charges) are plotted and anaysed. The categorical columns have count graphs plotted, whereas numeric columns have distribution graphs plotted.

DESIGN AND IMPLEMENTATION

Data Collection & Analysis

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
```

DESIGN AND IMPLEMENTATION

Data Collection & Analysis

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

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

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

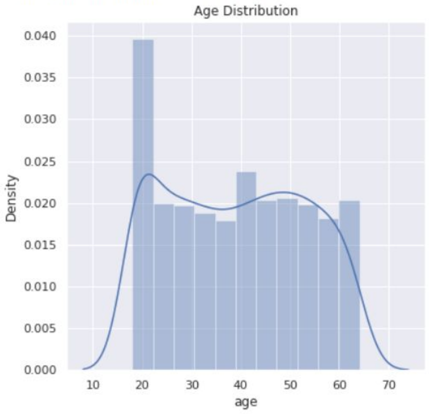
	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	13270.422265
std	14.049960	6.098187	1.205493	12110.011237
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.296250	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	9382.033000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

DESIGN AND IMPLEMENTATION

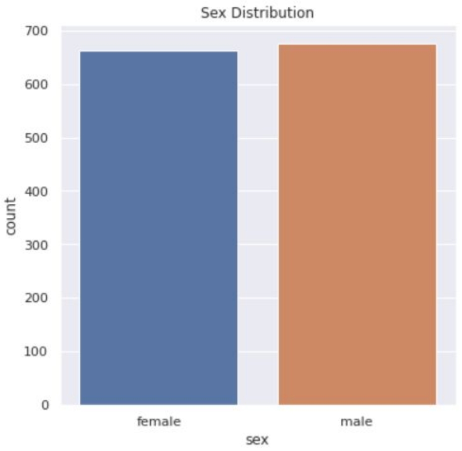
Data Collection & Analysis

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

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
```




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

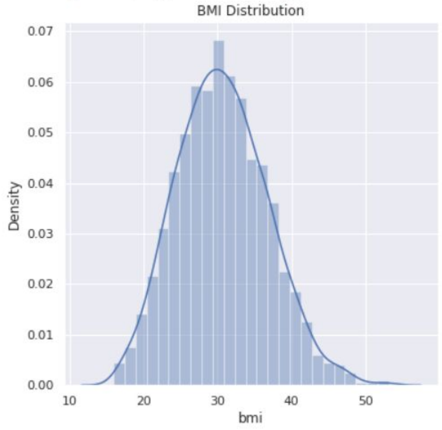


DESIGN AND IMPLEMENTATION

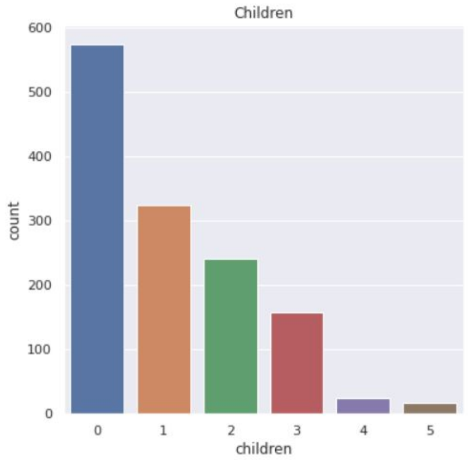
Data Collection & Analysis

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

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: Fut
  warnings.warn(msg, FutureWarning)
```



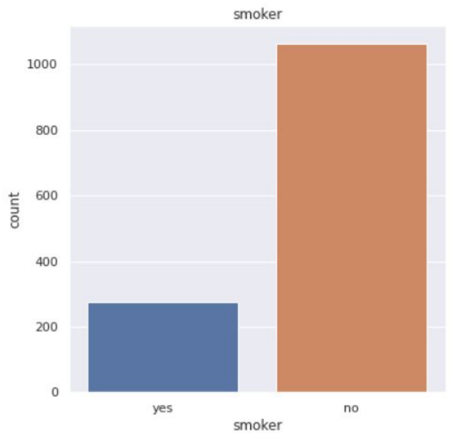

```
[ ]  # children column
     plt.figure(figsize=(6,6))
     sns.countplot(x='children', data=insurance_dataset)
     plt.title('Children')
     plt.show()
```



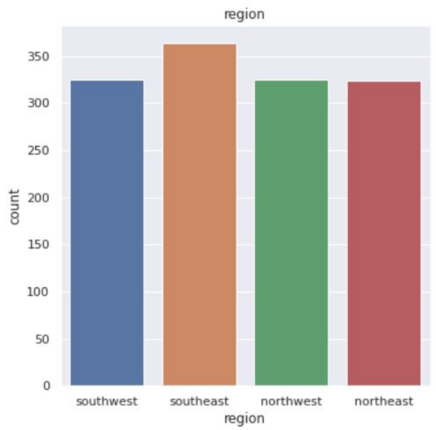
DESIGN AND IMPLEMENTATION

Data Collection & Analysis

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




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

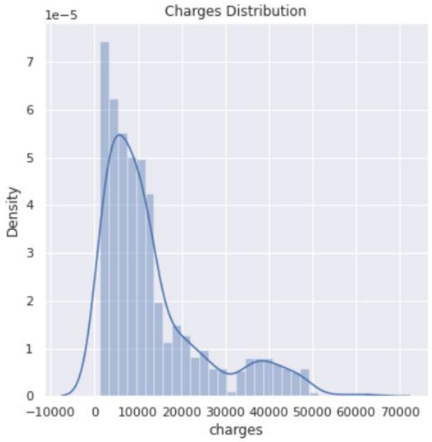


DESIGN AND IMPLEMENTATION

Data Collection & Analysis

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

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributi
  warnings.warn(msg, FutureWarning)
```



DESIGN AND IMPLEMENTATION

Data Pre-Processing:

- First we encode the categorical features because the data cannot be predicted as its not numerical, hence it needs to be encoded for further processing. The categorical features have a certain number of values hence the particular can be encoded as certain numbers.
- Since the charges column is the target of the predictive algorithm ,we split dataset into features and target for further processing. The other columns are the features.
- Then we divide the dataset further into train and test data in order to compare predicted insurance cost from the model with actual costs to evaluate the accuracies of the model.

DESIGN AND IMPLEMENTATION

Data Pre-Processing:

Encoding the categorical features

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

     # 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)
```

DESIGN AND IMPLEMENTATION

Data Pre-Processing:

Splitting the Features and Target

```
[20] 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]
```

DESIGN AND IMPLEMENTATION

Data Pre-Processing:

Splitting the data into Training data & Testing Data

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

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

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

DESIGN AND IMPLEMENTATION

Model Training

- We load the Linear Regression model on a variable.
- We then fit the X and Y training data on the X and Y axis of the Regression Model Respectively.

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

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

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

DESIGN AND IMPLEMENTATION

Model Evaluation

- To test or evaluate the data, we apply linear regression prediction on the X training data, so that its results can be compared to the actual Y training data.
- To compare the two, the R squared value is computed.
- R-squared is a statistical measure of how close the data are to the fitted regression line. It provides accuracy information by comparing the outputted data with the data provided.
- This same process is repeated for the Test Values.

DESIGN AND IMPLEMENTATION

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 :  0.7447273869684077
```

DESIGN AND IMPLEMENTATION

Building a predictive system

- We have first selected data to input.
- The input consists of all values for all columns except for the charges as that is our target.
- We have to make sure that the input for the encoded categorical columns are also encoded, or else, the predictive model will give an error.
- Since the input selected is stored in a parentheses, it is a tuple.
- We make it into a NumPy array so that it is easier to process.
- Then we need to reshape the array.
- This is done because the model does not know that we are predicting for one data point, as we have used over 1000 data points while training the model, and the model would expect the same number of values as the number of training data points used.
- That is why we use reshape(1,-1), as it tells the model to predict for one data point.

DESIGN AND IMPLEMENTATION

Building a predictive system

- Now we apply the prediction model to the reshaped input data.
- The predicted price of the insurance is displayed.
- We mention prediction[0], as prediction is in the form of a list which contains only one value, and 0 is the first index of the list, which we have to output.

```
[ ] 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  3760.0805764960587
```

CONCLUSION

In this project, The health insurance data was used to develop the regression model, and the predicted insurance premium from the model were compared with actual premiums to compare the accuracies of the model. It has been concluded that the algorithm is extremely accurate. For more accuracy in the future, a variety of multiple models can be used. The model can be applied to the data collected in coming years to predict the premium. This can help not only people but also insurance companies to work in tandem for better and more health centric insurance amount.

DATA MINING

PROJECT

References

- https://www.ijert.org/health-insurance-amount-prediction
- https://www.geeksforgeeks.org/ml-linear-regression/
- https://www.w3schools.com/python/python_ml_linear_regression.asp
- https://www.kaggle.com/mirichoi0218/insurance
- https://www.techscience.com/cmc/v70n2/44663/pdf

THANKS!