Insurance cost prediction

Submitted By

1) Sumit Kadam (asumitkadam@gmail.com)
2) Sakshi Kaulwar (kaulwarsakshi@gmail.com)
3) Shreyas Joshi (shreyassj15@gmail.com)
4) Abhishek kumar (s18_kumar_abhishek@gmail.com)

Under the Guidance of

Mr. Guruvansh Singh

M. Tech

Knowledge Solutions India

Contents

Figures and Graphs

ABSTRACT

The project is designed to predict the insurance cost accurately using various machine learning algorithms. It takes the Age, Sex, BMI, #Children, Smoker, Region, Charges as inputs from patient data and provides the output Insurance Cost. This type of modelling will add value to various insurance organisations and will aid in budgeting and with

We have implemented this project using different algorithms such as Multiple Linear Regression, Random Forest, PCA with Multiple Linear Regression, PCA with Random Forest. These algorithms provide the optimum accuracy for the model.

MULTIPLE LINEAR REGRESSION

Multiple linear regression (MLR), also known simply as multiple regression, is a statistical technique that uses several explanatory variables to predict the outcome of a response variable. Multiple regression is an extension of linear (OLS) regression that uses just one explanatory variable.

A simple linear regression is a function that allows an analyst or statistician to make predictions about one variable based on the information that is known about another variable. Linear regression can only be used when one has two continuous variables—an independent variable and a dependent variable. The independent variable is the parameter that is used to calculate the dependent variable or outcome. A multiple regression model extends to several explanatory variables.

At the centre of the multiple linear regression analysis is the task of fitting a single line through a scatter plot. More specifically the multiple linear regression fits a line through a multi-dimensional space of data points. The simplest form has one dependent and two independent variables. The dependent variable may also be referred to as the outcome variable or regressand. The independent variables may also be referred to as the predictor variables or repressor.

There are 3 major uses for multiple linear regression analysis. First, it might be used to identify the strength of the effect that the independent variables have on a dependent variable.

Second, it can be used to forecast effects or impacts of changes. That is, multiple linear regression analysis helps us to understand how much will the dependent variable change when we change the independent variables. For instance, a multiple linear regression can tell you how much GPA is expected to increase (or decrease) for every one-point increase (or decrease) in IQ

Third, multiple linear regression analysis predicts trends and future values. The multiple linear regression analysis can be used to get point estimates.

MLR can result in simple equations that can be used for quantitation. For example, the constituents of food, such as moisture and fat, can be determined using spectroscopic data at a few NIR wavelengths. The disadvantages of MLR are that it requires the operator to select the wavelengths. The selection of inappropriate wavelengths can result in poor models that are mathematically unstable.

The multiple regression model is based on the following assumptions:

- There is a linear relationship between the dependent variables and the independent variables.
- The independent variables are not too highly correlated with each other.
- yi observations are selected independently and randomly from the population.
- Residuals should be normally distributed with a mean of 0 and variance σ.

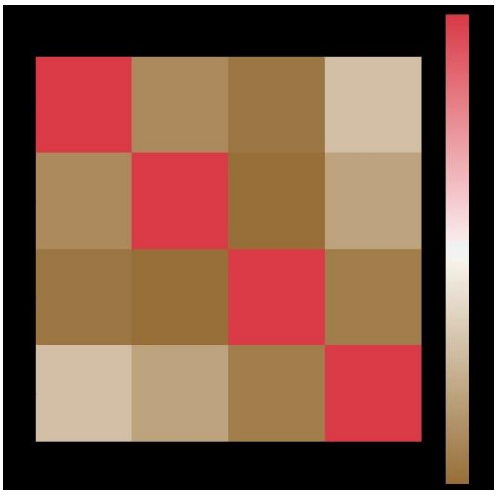
The coefficient of determination (R-squared) is a statistical metric that is used to measure how much of the variation in outcome can be explained by the variation in the independent variables. R^2 always increases as more predictors are added to the MLR model even though the predictors may not be related to the outcome variable.

R^2 by itself can't thus be used to identify which predictors should be included in a model and which should be excluded. R^2 can only be between 0 and 1, where 0 indicates that the outcome cannot be predicted by any of the independent variables and 1 indicates that the outcome can be predicted without error from the independent variables.

When interpreting the results of a multiple regression, beta coefficients are valid while holding all other variables constant ("all else equal"). The output from a multiple regression can be displayed horizontally as an equation, or vertically in table form.

LIBRARIES USED:

from sklern.linear_model import Linear regression.



Graph Of Multiple Linear Regression

RANDOM FOREST

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

The general method of random decision forests was first proposed by Ho in 1995. Ho established that forests of trees splitting with oblique hyperplanes can gain accuracy as they grow without suffering from overtraining, as long as the forests are randomly restricted to be sensitive to only selected feature dimensions. A subsequent work along the same lines concluded that other splitting methods behave similarly, as long as they are randomly forced to be insensitive to some feature dimensions. Note that this observation of a more complex classifier (a larger forest) getting more accurate nearly monotonically is in sharp contrast to the common belief that the complexity of a classifier can only grow to a certain level of accuracy before being hurt by overfitting. The explanation of the forest method's resistance to overtraining can be found in Kleinberg's theory of stochastic discrimination.

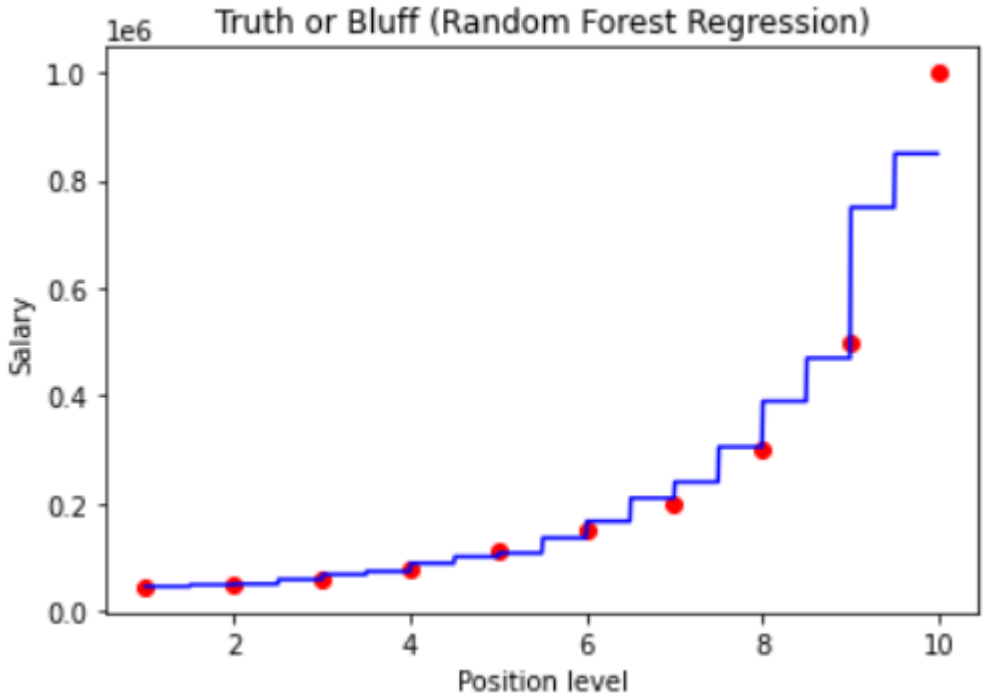
Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model’s prediction.

The fundamental concept behind random forest is a simple but powerful one — the wisdom of crowds.

The low correlation between models is the key. Just like how investments with low correlations (like stocks and bonds) come together to form a portfolio that is greater than the sum of its parts, uncorrelated models can produce ensemble predictions that are more accurate than any of the individual predictions. The reason for this wonderful effect is that the trees protect each other from their individual errors (as long as they don’t constantly all err in the same direction). While some trees may be wrong, many other trees will be right, so as a group the trees are able to move in the correct direction. So the prerequisites for random forest to perform well are:

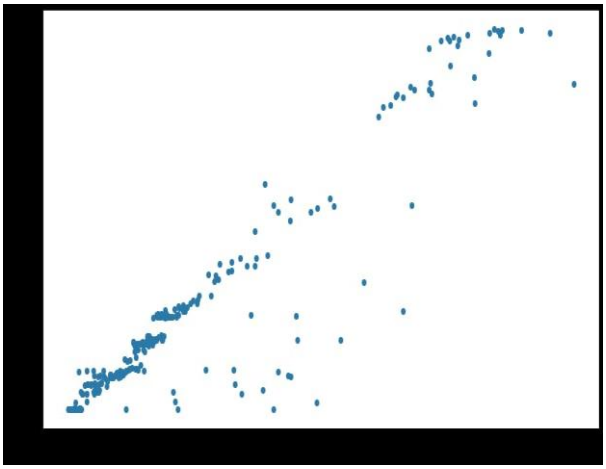
1. There needs to be some actual signal in our features so that models built using those features do better than random guessing.

2. The predictions (and therefore the errors) made by the individual trees need to have low correlations with each other.



Predicting and representing the data

LIBRARY USED:-from sklern.ensemble import RandomForest Regressor.



Graph of Random forest Regression

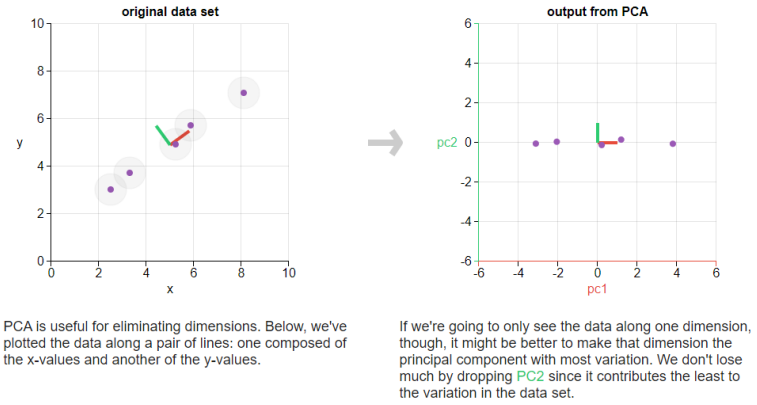
PCA WITH MLR:

PCA makes maximum variability in the dataset more visible by rotating the axes. PCA identifies a list of the principal axes to describe the underlying dataset before ranking them according to the amount of variance captured by each.

Principal component analysis (PCA) is a technique used to emphasize variation and bring out strong patterns in a dataset. It's often used to make data easy to explore and visualize.

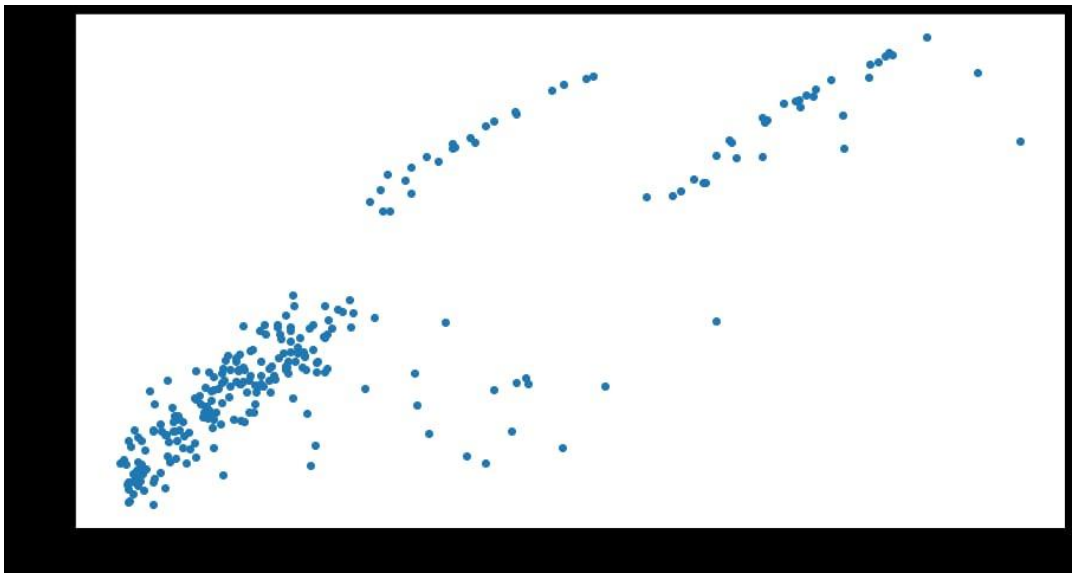
First, consider a dataset in only two dimensions, like (height, weight). This dataset can be plotted as points in a plane. But if we want to tease out variation, PCA finds a new coordinate system in which every point has a new (x,y) value. The axes don't actually mean anything physical; they're combinations of height and weight called "principal components" that are chosen to give one axes lots of variation.

Drag the points around in the following visualization to see PC coordinate system adjusts.



Graph Of PCA

LIBRARY USED: - from sklern.decomposition import PCA
from sklern.linear_model import Linear regression.



Graph Of PCA With Multiple Linear Regression

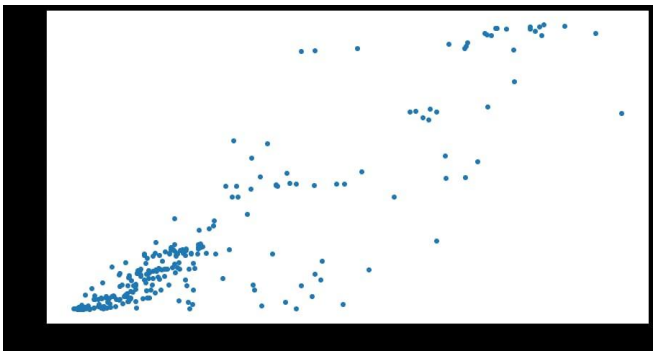
PCA WITH RFR:

PCA performs dimensionality reduction, which can reduce the number of features for the Random Forest to process, so PCA might help speed up the training of your Random Forest model. Note that computational cost is one of the biggest drawbacks of Random Forests (it can take a long time to run the model). PCA can become really important especially when you are working with hundreds or even thousands of predicting features. So if the most important thing is to simply have the best performing model, and interpreting feature importance can be sacrificed, then PCA may be useful to try.

.

LIBRARY USED:- from sklern.decomposition import PCA

from sklern.ensemble importRandomForestRegressor.



Graph Of PCA with Random Forest

Conclusion:

We have found That, we can predict the cost of insurance with the accuracy of 80% which is quite enough. Now we can predict any one's insurance cost using this machine learning model on the basis of the database we have!

CODE

```
import numpy as np # linear algebra
import pandas as pd # data processing


# for data visualizations
import matplotlib.pyplot as plt
import seaborn as sns


# reading the data


data = pd.read_csv(r'C:\Users\Sumit\Desktop\insurance.csv')


# checking the shape
print(data.shape)


# checking the head of the dataset


data.head()
```

```
# describing the data

data.describe()

# checking if the dataset contains any NULL values

data.isnull().any()

sns.pairplot(data)

# lmplot between age and charges

sns.lmplot('age', 'charges', data = data)

# bubble plot to show relation bet age, charges and children

plt.rcParams['figure.figsize'] = (15, 8)
plt.scatter(x = data['age'], y = data['charges'], s = data['children']*100, alpha = 0.2, color = 'red')
plt.title('Bubble plot', fontsize = 30)
plt.xlabel('Age')
plt.ylabel('Charges')
plt.legend()
plt.show()
```

```
# plotting the correlation plot for the dataset

f, ax = plt.subplots(figsize = (10, 10))

corr = data.corr()
sns.heatmap(corr, mask = np.zeros_like(corr, dtype = np.bool),
            cmap = sns.diverging_palette(50, 10, as_cmap = True), square = True, ax = ax)

# removing unnecassary columns from the dataset

data = data.drop('region', axis = 1)

print(data.shape)

data.columns

# label encoding for sex and smoker

# importing label encoder
from sklearn.preprocessing import LabelEncoder

# creating a label encoder
le = LabelEncoder()
```

```
# label encoding for sex
# 0 for females and 1 for males
data['sex'] = le.fit_transform(data['sex'])

# label encoding for smoker
# 0 for smokers and 1 for non smokers
data['smoker'] = le.fit_transform(data['smoker'])

# splitting the dependent and independent variable

x = data.iloc[:,:5]
y = data.iloc[:,5]

print(x.shape)
print(y.shape)

# splitting the dataset into training and testing sets

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 30)

print(x_train.shape)
```

```
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)


# standard scaling


from sklearn.preprocessing import StandardScaler


# creating a standard scaler
sc = StandardScaler()


# feeding independents sets into the standard scaler
x_train = sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)


# Set data
from math import pi
df = pd.DataFrame({
'group': [i for i in range(0, 1338)],
'Age': data['age'],
'Charges': data['charges'],
'Children': data['children'],
'BMI': data['bmi']
})


# importing the model
```

```
# Multiple linear regression
from sklearn.linear_model import LinearRegression
model = LinearRegression()


# Fit linear model by passing training dataset
model.fit(x_train,y_train)


# Predicting the target variable for test datset
predictions = model.predict(x_test)


print('THE PREDICTION VALUE IN MULTIPLE LINEAR REGRESSOR IS:\n')
print(predictions)


#plotting the y prediction
import matplotlib.pyplot as plt
plt.scatter(y_test,predictions)
plt.title('Multiple Linear Regression')
plt.xlabel('Y Test')
plt.ylabel('Predicted Y')
plt.show()


# REGRESSION ANALYSIS
# RANDOM FOREST


from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score
```

```
# creating the model
model = RandomForestRegressor(n_estimators = 40, max_depth = 4, n_jobs = -1)


# feeding the training data to the model
model.fit(x_train, y_train)


# predicting the test set results
y_pred = model.predict(x_test)
print('THE PREDICTION VALUE OF IN RANDOM FOREST REGRESSOR IS:\n')
print(y_pred)


#plotting the y prediction
import matplotlib.pyplot as plt
plt.scatter(y_test,y_pred)
plt.title('Random Forest Regression')
plt.xlabel('Y Test')
plt.ylabel('Predicted Y')
plt.show()


# feature extraction


from sklearn.decomposition import PCA


pca = PCA(n_components = None)
```

```
x_train = pca.fit_transform(x_train)
x_test = pca.transform(x_test)


# importing the model
# Multiple linear regression
from sklearn.linear_model import LinearRegression
model = LinearRegression()


# Fit linear model by passing training dataset
model.fit(x_train,y_train)


# Predicting the target variable for test datset
predictions = model.predict(x_test)


print('THE PREDICTION VALUE OF PCA WITH MULTIPLE LINEAR REGRESSOR IS:\n')
print(predictions)


#plotting the y prediction
import matplotlib.pyplot as plt
plt.scatter(y_test,predictions)
plt.title('PCA with Multiple Linear Regression')
plt.xlabel('Y Test')
plt.ylabel('Predicted Y')
plt.show()
```

```
# feature extraction

from sklearn.decomposition import PCA

pca = PCA(n_components = None)

x_train = pca.fit_transform(x_train)
x_test = pca.transform(x_test)

# REGRESSION ANALYSIS
# RANDOM FOREST

from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score

# creating the model
model = RandomForestRegressor(n_estimators = 40, max_depth = 4, n_jobs = -1)

# feeding the training data to the model
model.fit(x_train, y_train)

# predicting the test set results
y_pred = model.predict(x_test)
```

```
print('THE PREDICTION VALUE OF IN PCA WITH RANDOM FOREST REGRESSOR IS:\n')
print(y_pred)


#plotting the y prediction
import matplotlib.pyplot as plt
plt.scatter(y_test,y_pred)
plt.title('PCA with Random Forest Regression')
plt.xlabel('Y Test')
plt.ylabel('Predicted Y')
plt.show()



# calculating the mean squared error
mse = np.mean((y_test - y_pred)**2, axis = None)
print("MSE :", mse)


# Calculating the root mean squared error
rmse = np.sqrt(mse)
print("RMSE :", rmse)


# Calculating the r2 score
r2 = r2_score(y_test, y_pred)
print("r2 score :", r2)
```