bhj0s8gbr

January 27, 2025

[ ]:
```
# PROJECT REPORT FOR RANDOM FOREST REGRESSION USING MEDICAL COST PERSONAL␣
 ↪DATASET:


#Project Title: Predicting Medical Costs using Random Forest Regression

#Dataset: Medical Cost Personal Dataset (Insurance Dataset)

#Dataset Characteristics:

#- Number of instances: 1338
#- Number of attributes: 7 (age, sex, bmi, children, smoker, region, charges)
#- Task: Regression (predicting medical costs)

#Objective:
#Predict medical costs based on patient characteristics using Random Forest␣
 ↪Regression.

#Methodology:

#1. Data Preprocessing:
   # - Loaded dataset and converted categorical variables into numerical␣
 ↪variables.
   # - Split data into training (80%) and testing sets (20%).
#2. Feature Scaling:
   # - Applied Min-Max Scaler to scale numerical features (age, bmi, children,␣
 ↪charges).
#3. Random Forest Regression:
    #- Trained Random Forest Regressor model with 100 estimators on training␣
 ↪data.
    #- Predicted medical costs on testing data.
#4. Evaluation Metrics:
  # Mean Squared Error: 0.005284974769387913
   #Mean Absolute Error: 0.039337624797441725
   #Root Mean Squared Error: 0.07269783194420527
   #R-squared: 0.8663906157908996
```

```
# Accuracy : 86.6%
```

[56]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import math
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import mean_squared_error,mean_absolute_error, r2_score
```

[57]:
```
#load the dataset
data = pd.read_csv(r"C:\Users\91703\Downloads\MEDICAL REPORT OF A PATIENT.csv")
```

[58]:
```
data.head()
```

[58]:

	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

[59]:
```
data.shape
```

[59]:
```
(1338, 7)
```

[60]:
```
data.describe()
```

[60]:

	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

[61]:
```
data.isnull().sum()
```

[61]:
```
age         0
sex         0
bmi         0
children    0
```

```
smoker      0
region      0
charges     0
dtype: int64
```

[62]:
```
data.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
```

[63]:
```
#LabelEncoder for multiple columns
columns_to_encode = ['sex', 'smoker', 'region']
for col in columns_to_encode:
    le = LabelEncoder()
    data[col] = le.fit_transform(data[col])
```

[64]:
```
data.head()
```

[64]:

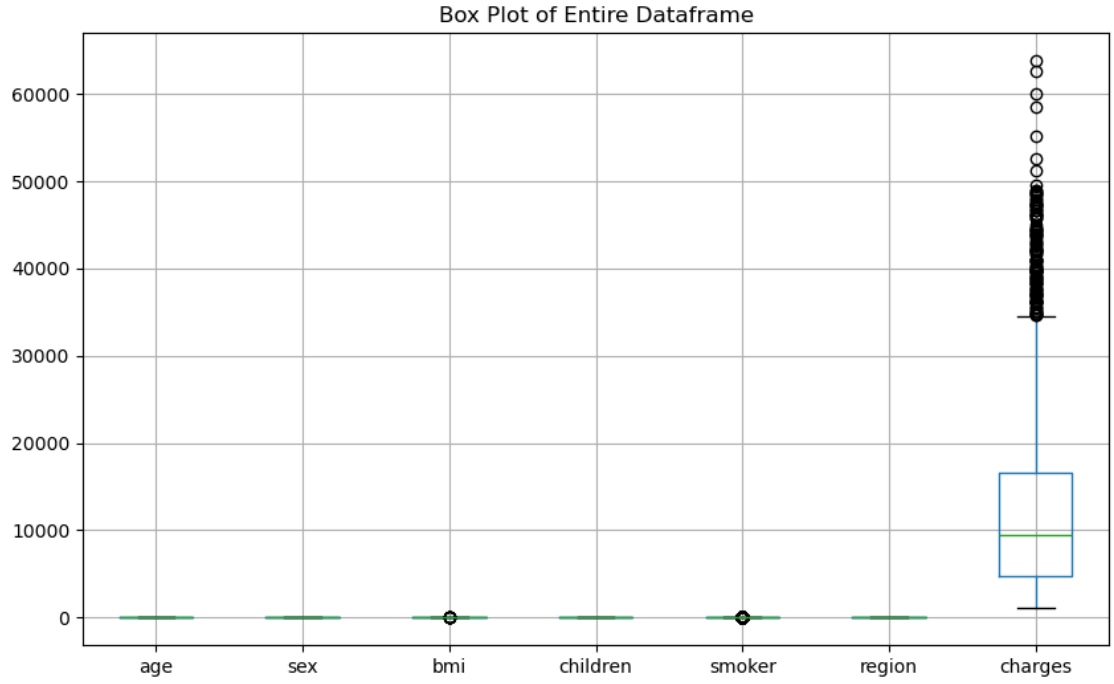
	age	sex	bmi	children	smoker	region	charges
0	19	0	27.900	0	1	3	16884.92400
1	18	1	33.770	1	0	2	1725.55230
2	28	1	33.000	3	0	2	4449.46200
3	33	1	22.705	0	0	1	21984.47061
4	32	1	28.880	0	0	1	3866.85520

[65]:
```
data.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   int32
 2   bmi       1338 non-null   float64
 3   children  1338 non-null   int64
```

```
 4   smoker    1338 non-null   int32
 5   region    1338 non-null   int32
 6   charges   1338 non-null   float64
dtypes: float64(2), int32(3), int64(2)
memory usage: 57.6 KB
```

[66]:
```
#now by using the boxplot we will find out the outliers
data.select_dtypes(include=['int64','int32','float64']).boxplot(figsize=(10,6))
plt.title('Box Plot of Entire Dataframe')
plt.show()
```



[ ]:
```

```

[67]:
```
#here we are using the min - max scaler to range the data into 0 to 1
scaler = MinMaxScaler()
data[['age', 'bmi', 'children', 'charges']] = scaler.fit_transform(data[['age',␣
 ↪'bmi', 'children', 'charges']])
```

[68]:
```
data.head()
```

[68]:

	age	sex	bmi	children	smoker	region	charges
0	0.021739	0	0.321227	0.0	1	3	0.251611
1	0.000000	1	0.479150	0.2	0	2	0.009636
2	0.217391	1	0.458434	0.6	0	2	0.053115
3	0.326087	1	0.181464	0.0	0	1	0.333010

```
4  0.304348    1  0.347592       0.0       0       1  0.043816
```

[69]:
```
# Define target variable (y) and feature variables (X)
y = data['charges']
X = data[['age','sex','bmi','children','smoker','region']]
```

[70]:
```
data.head()
```

[70]:

	age	sex	bmi	children	smoker	region	charges
0	0.021739	0	0.321227	0.0	1	3	0.251611
1	0.000000	1	0.479150	0.2	0	2	0.009636
2	0.217391	1	0.458434	0.6	0	2	0.053115
3	0.326087	1	0.181464	0.0	0	1	0.333010
4	0.304348	1	0.347592	0.0	0	1	0.043816

[71]:
```
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[72]:
```
model = RandomForestRegressor()
model.fit(X_train, y_train)
```

[72]:
```
RandomForestRegressor()
```

[73]:
```
# Make predictions on test data
y_pred = model.predict(X_test)
print(y_pred)
```

```
[0.15533068 0.06339023 0.4313026  0.14201806 0.53528662 0.12775321
 0.01733873 0.21135977 0.09555137 0.1518983  0.29400872 0.10143351
 0.05222588 0.71443541 0.75227557 0.70173465 0.14084863 0.66354271
 0.12428188 0.36167986 0.06657798 0.13963013 0.0284154  0.03674033
 0.17366091 0.16849366 0.19183281 0.11138572 0.14020921 0.05628914
 0.10738079 0.17538469 0.02821113 0.08500017 0.039108   0.16056365
 0.01919187 0.10723362 0.36081703 0.61367868 0.09812775 0.05112719
 0.17701488 0.20213709 0.08148889 0.21729682 0.26105588 0.06808051
 0.65377079 0.08965878 0.2044503  0.01374952 0.09680285 0.01134343
 0.1761083  0.16033434 0.04167859 0.68596582 0.18351906 0.20460639
 0.20111005 0.09636121 0.3415675  0.10667063 0.16241698 0.0733962
 0.27278062 0.2233442  0.10333714 0.01529718 0.0996729  0.15137978
 0.13459855 0.14927854 0.11423014 0.05987438 0.11069482 0.19498014
 0.05046718 0.14983198 0.00592874 0.4332361  0.07442801 0.58199686
 0.63057267 0.64201402 0.07549677 0.1960528  0.13024513 0.2156375
 0.32962389 0.40668389 0.43017583 0.07317222 0.71061422 0.1045832
 0.30098417 0.01991412 0.26268688 0.16516046 0.05609208 0.01065911
 0.08029381 0.23524564 0.20993078 0.10675367 0.11824608 0.35246978
 0.00932371 0.36479957 0.00258545 0.03659273 0.18086252 0.62174913
```

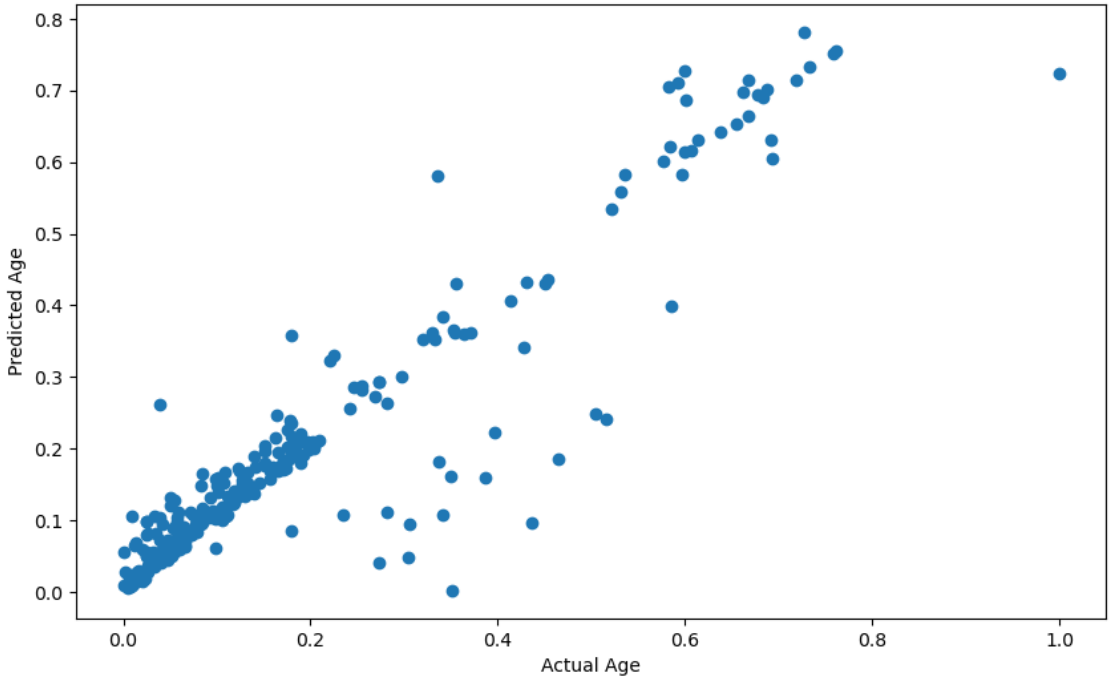
```
0.14572192 0.06811057 0.18590677 0.38364546 0.09533923 0.08251203
0.11163703 0.1583183  0.24660658 0.01770548 0.10492423 0.10074395
0.14837329 0.15578686 0.23897292 0.0644445  0.0462451  0.0931273
0.08387135 0.16695097 0.08041826 0.3589921  0.08640504 0.58061959
0.71424059 0.60191798 0.06426538 0.18909026 0.02708073 0.2407472
0.02124232 0.3622516  0.07550053 0.06016488 0.16974712 0.06466857
0.78157116 0.07919918 0.0097538  0.70437338 0.09747341 0.07218979
0.19893701 0.14103304 0.55806211 0.61521716 0.20856661 0.0563447
0.32359346 0.06036902 0.04774941 0.10417493 0.75523062 0.69363893
0.63121161 0.09770078 0.15664741 0.09956984 0.10788339 0.05299244
0.02970839 0.35277287 0.25697537 0.20896829 0.24928387 0.17173881
0.43638145 0.10649678 0.12210494 0.0849414  0.09074172 0.04991407
0.0947115  0.07303108 0.12448121 0.17934365 0.04559369 0.06080746
0.02774927 0.7266263  0.1919631  0.13318444 0.0502405  0.22561316
0.01796577 0.16881555 0.03611493 0.36180066 0.12159812 0.10848702
0.28133866 0.28599396 0.17283305 0.13157558 0.1522659  0.04984121
0.22120875 0.18168563 0.16631042 0.28793141 0.11343324 0.04524158
0.06150246 0.2076066  0.20914435 0.06993407 0.01588054 0.13147301
0.11774856 0.69679636 0.01511005 0.58341676 0.00997076 0.01390568
0.14094808 0.15788579 0.02046751 0.13920313 0.07023265 0.39956363
0.17506189 0.13308582 0.05771047 0.09492113 0.68946245 0.02417048
0.19088417 0.60538646 0.04002826 0.04092651 0.00840934 0.02960478
0.04815847 0.05508764 0.20038698 0.01777028 0.02027302 0.10747623
0.05574808 0.17252394 0.09897403 0.04082364 0.18615398 0.08111801
0.13150063 0.10143324 0.10752752 0.20918569 0.29371001 0.73303697
0.18544697 0.08383494 0.72428629 0.13797056]
```

[74]:
```
# Evaluate model performance
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
rmse = math.sqrt(mse)
r2 = r2_score(y_test, y_pred)
print("Mean Squared Error:", mse)
print("Mean Absolute Error:", mae)
print("Root Mean Squared Error:", rmse)
print("R-squared:", r2)
```

```
Mean Squared Error: 0.005284974769387913
Mean Absolute Error: 0.039337624797441725
Root Mean Squared Error: 0.07269783194420527
R-squared: 0.8663906157908996
```

[75]:
```
# Plot predicted vs actual values
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Age")
plt.ylabel("Predicted Age")
```

```
plt.show()
```


Predicted Age
Actual Age
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.0
0.2
0.4
0.6
0.8
1.0