# Insurance Cost Analysis

Aman Bhattarai

## Table of contents

## 1 Introduction

This project analyzes an insurance database to understand factors affecting insurance charges and develop predictive models.

# 2 Setup

```python
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression, Ridge
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler, PolynomialFeatures
import matplotlib.pyplot as plt
import seaborn as sns

# Set style and size
plt.rcParams['figure.figsize'] = (10, 6)
```

# 3 Data Loading and Cleaning

## 3.1 Import Dataset

```python
# Load the dataset
filepath = 'medical_insurance_dataset.csv'
df = pd.read_csv(filepath, header=None)

# Add headers to the dataframe
headers = ["age", "gender", "bmi", "no_of_children", "smoker", "region", "charges"]
df.columns = headers

# Display the first 10 rowsFirst 10 rows of the dataset:")
print(df.head(10))
```

```
   age  gender     bmi  no_of_children smoker  region      charges
0   19       1  27.900               0      1       3  16884.92400
1   18       2  33.770               1      0       4   1725.55230
2   28       2  33.000               3      0       4   4449.46200
3   33       2  22.705               0      0       1  21984.47061
4   32       2  28.880               0      0       1   3866.85520
5   31       1  25.740               0      ?       4   3756.62160
6   46       1  33.440               1      0       4   8240.58960
```

```
7  37        1  27.740              3      0       1   7281.50560
8  37        2  29.830              2      0       2   6406.41070
9  60        1  25.840              0      0       1  28923.13692
```

### 3.2 Handle Missing Data

```
# Replace '?' with NaN
df.replace('?', np.nan, inplace=True)

print("Missing values:")
print(df.isnull().sum())

# Handle missing data
mean_age = df['age'].astype('float').mean()
df['age'] = df['age'].replace(np.nan, mean_age)

is_smoker = df['smoker'].value_counts().idxmax()
df['smoker'] = df['smoker'].replace(np.nan, is_smoker)

# Update data types
df[["age", "smoker"]] = df[["age", "smoker"]].astype("int")

# Round charges to 2 decimal places
df[["charges"]] = np.round(df[["charges"]], 2)

print("\nDataframe info after cleaning:")
print(df.info())
```

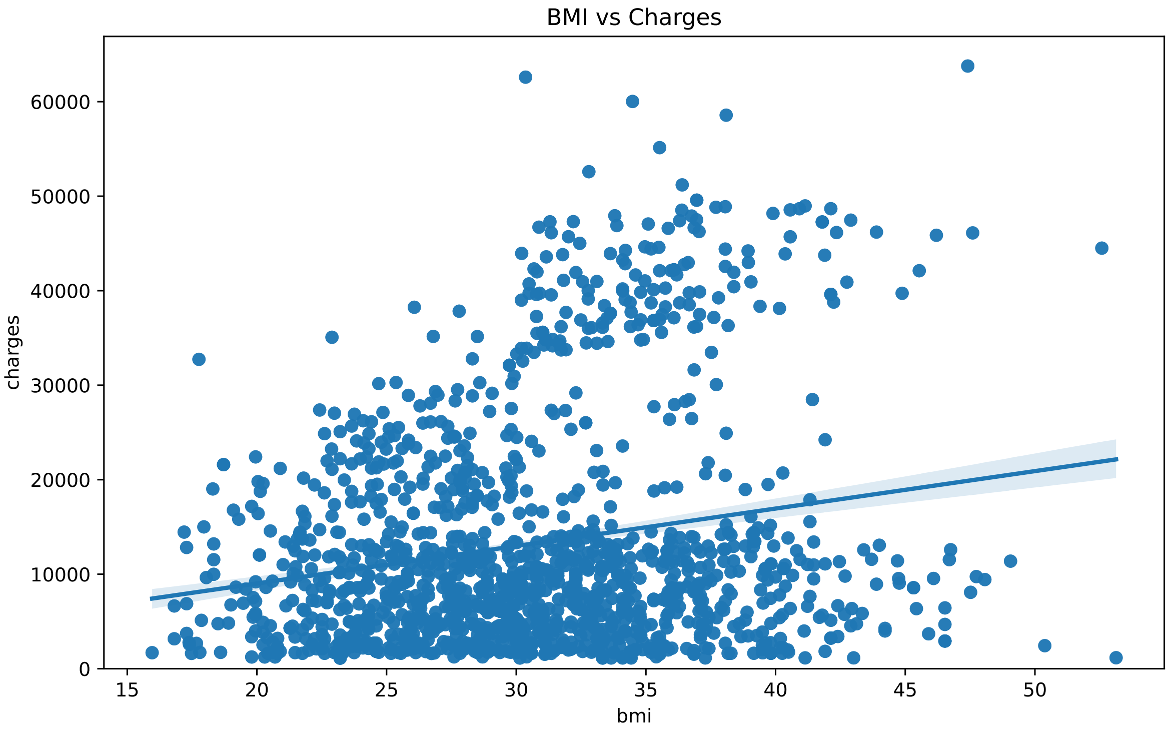
```
Missing values:
age               4
gender            0
bmi               0
no_of_children    0
smoker            7
region            0
charges           0
dtype: int64

Dataframe info after cleaning:
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 2772 entries, 0 to 2771
Data columns (total 7 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             2772 non-null   int64
 1   gender          2772 non-null   int64
 2   bmi             2772 non-null   float64
 3   no_of_children  2772 non-null   int64
 4   smoker          2772 non-null   int64
 5   region          2772 non-null   int64
 6   charges         2772 non-null   float64
dtypes: float64(2), int64(5)
memory usage: 151.7 KB
None
```

# 4 Exploratory Data Analysis (EDA)

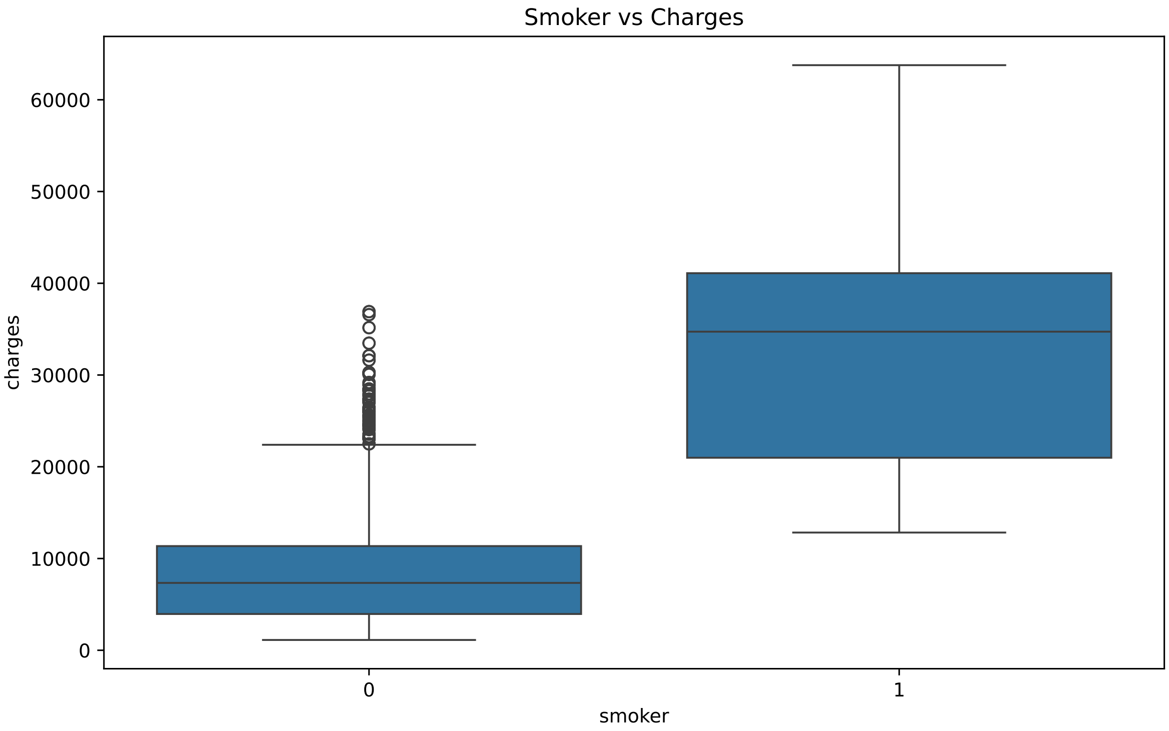
## 4.1 Regression Plot: BMI vs Charges

```python
plt.figure(figsize=(10, 6))
sns.regplot(x="bmi", y="charges", data=df)
plt.ylim(0,)
plt.title("BMI vs Charges")
plt.show()
```



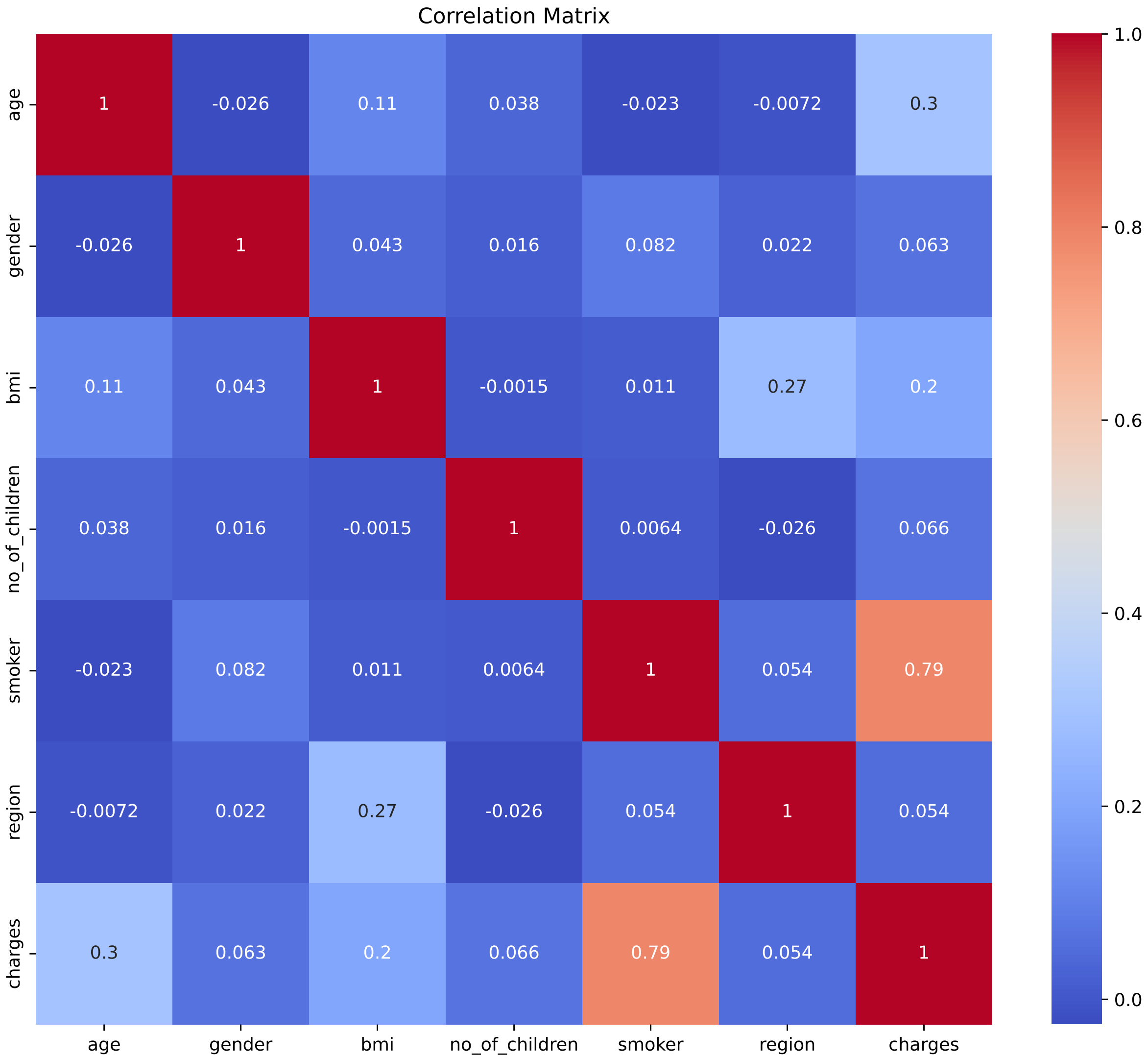
## 4.2 Box Plot: Smoker vs Charges

```
plt.figure(figsize=(10, 6))
sns.boxplot(x='smoker', y='charges', data=df)
plt.title("Smoker vs Charges")
plt.show()
```



## 4.3 Correlation Matrix

```
plt.figure(figsize=(12, 10))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title("Correlation Matrix")
plt.show()
```



# 5 Model Development

## 5.1 Linear Regression: Smoker Only

```
lr = LinearRegression()
X = df[['smoker']]
Y = df[['charges']]
lr.fit(X, Y)
print(f"R2 score (smoker only): {lr.score(X, Y):.4f}")
```

```
R2 score (smoker only): 0.6222
```

### 5.2 Linear Regression: All Attributes

```
Xnew = df.drop('charges', axis=1)
lr.fit(Xnew, Y)
print(f"R2 score (all attributes): {lr.score(Xnew, Y):.4f}")
```

```
R2 score (all attributes): 0.7504
```

### 5.3 Pipeline: StandardScaler, PolynomialFeatures, LinearRegression

```
Input = [('scale', StandardScaler()),
         ('polynomial', PolynomialFeatures(include_bias=False)),
         ('model', LinearRegression())]
pipe = Pipeline(Input)
Xupd = Xnew.astype(float)
pipe.fit(Xupd, Y)
ypipe = pipe.predict(Xupd)
print(f"R2 score (pipeline): {r2_score(Y, ypipe):.4f}")
```

```
R2 score (pipeline): 0.8453
```

## 6 Model Refinement

### 6.1 Data Splitting

```
x_train, x_test, y_train, y_test = train_test_split(Xupd, Y, test_size=0.2, random_state=1)
```

### 6.2 Ridge Regression

```
RR = Ridge(alpha=0.1)
RR.fit(x_train, y_train)
y_predict = RR.predict(x_test)
print(f"R2 score (Ridge Regression): {r2_score(y_test, y_predict):.4f}")
```

```
R2 score (Ridge Regression): 0.6761
```

### 6.3 Polynomial Ridge Regression

```
pr = PolynomialFeatures(degree=2)
x_train_pr = pr.fit_transform(x_train)
x_test_pr = pr.fit_transform(x_test)
RR.fit(x_train_pr, y_train)
y_predict_pr = RR.predict(x_test_pr)
print(f"R2 score (Polynomial Ridge Regression): {r2_score(y_test, y_predict_pr):.4f}")
```

```
R2 score (Polynomial Ridge Regression): 0.7836
```

## 7 Conclusion

This analysis explored various factors affecting insurance charges and developed predictive models using different regression techniques. The pipeline regression model showed the best performance in predicting insurance charges based on the given attributes.