PDAN POE PART 1

Analysis Report

APRIL 25, 2025

ST10059601 – TRISTAN GULZAR

Contents

Introduction

Problem Statement

Medical aid providers aim to offer fair, transparent pricing models based on individual risk profiles. This project builds a linear regression model to predict medical charges based on demographic and lifestyle data—age, sex, BMI, smoking status, number of children, and region. Though the data is US-based, the insights and methodology are valuable for application in South African contexts as a proof of concept.

Dataset Overview

Source of the Data

The dataset is available from Kaggle and is used for insurance pricing and health cost prediction exercises. It was originally compiled to study how personal habits and demographics affect healthcare costs.

Explanation of the Features

Here's a breakdown of the dataset's key features:

- **Age**: Numeric value indicating the person's age.
- **Sex**: Gender – male or female.
- **BMI**: Body Mass Index – a measure of weight adjusted for height.
- **Children**: Number of children the person has, which could affect family-related health costs.
- **Smoker**: Whether the person is a smoker (yes or no).
- **Region**: The person's geographical region in the US.
- **Charges**: The total medical cost billed to the insurer. This is the value we want to predict.

Target Variable

The target variable is charges, a continuous numeric value representing the total medical expenses. This value is what our model is trained to predict based on the other features.

Loading and Inspecting the Dataset

We first import all the necessary libraries that we are going to work with in this project:

```
# Import necessary libraries for data manipulation, visualisation, and modeling

import pandas as pd
import numpy as np
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.metrics import mean_absolute_error, mean_squared_error, r2_score # Evaluation metrics for regression performance
from sklearn.preprocessing import OneHotEncoder
```

We then start loading the dataset using the pandas library:

df = pd.read_csv("insurance.csv")

“df” is the variable name in which we will store the dataset in order to manipulate the values.

Once loaded we then conduct an initial inspection.

Initial Inspection

- df.head() – this code simply displays the first five rows of the dataset and helps us get an initial understanding of the structure and content of the data
- df.info() – this code provides the total number of entries in the dataset and information about the columns such as the column names and their data types.

```
<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
```

- df.describe() – this code provides a basic statistic about the dataset. It helps summarize the distribution and spread of the numerical features. With this information we can then decide if there are any outliers and if they would need to be removed. In this dataset, I noticed that BMI and Charges had outliers but chose not to remove them as these are not data entry errors but represent real, expensive cases.

	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

- df.isnull().sum() – this code checks to see if there are any null values in the dataset. We need to do this as null values could affect our model negatively which can cause skewed predictions.

Why did we do these steps?

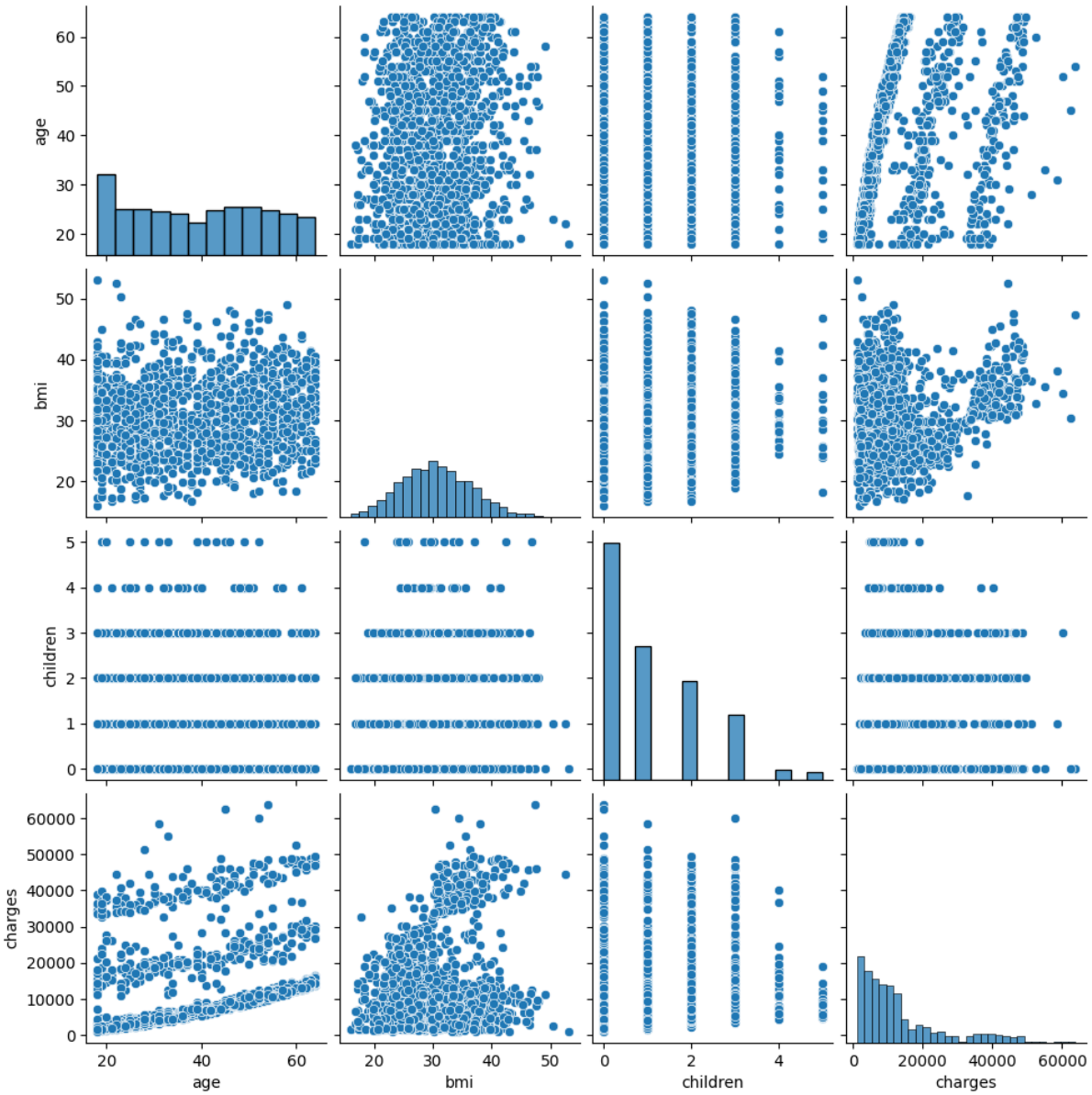
Data must be clean and structured before modeling to prevent runtime errors and inaccurate predictions (Tableau, 2024). Ensuring correct data types and absence of nulls supports effective preprocessing and modeling.

Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is a step in Data Science projects which involve looking and visualising data to help discover patterns, how different parts of data is connected and its main features (Geeksforgeeks, 2025).

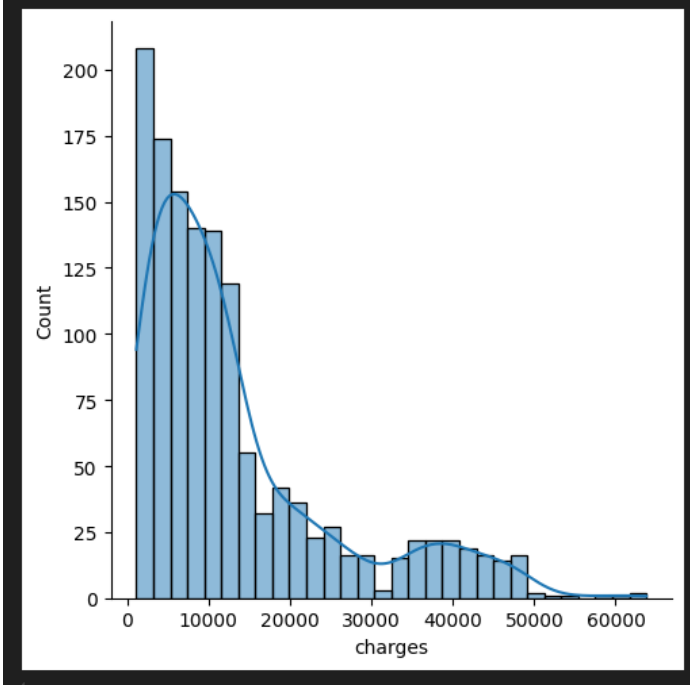
The following are different visualisation steps I did in order to recognize the features mentioned above:

- **Pairplot** – this graph was created to visually explore relationships between numerical features.



- **Displot against charges** – this plot shows the distribution of charges. From this graph, I was able to determine that the graph was right skewed and also helped further prove my statement that there are outliers.

```
sns.displot(df['charges'], kde=True)
✓ 0.3s
<seaborn.axisgrid.FacetGrid at 0x22ac3ff7230>
```

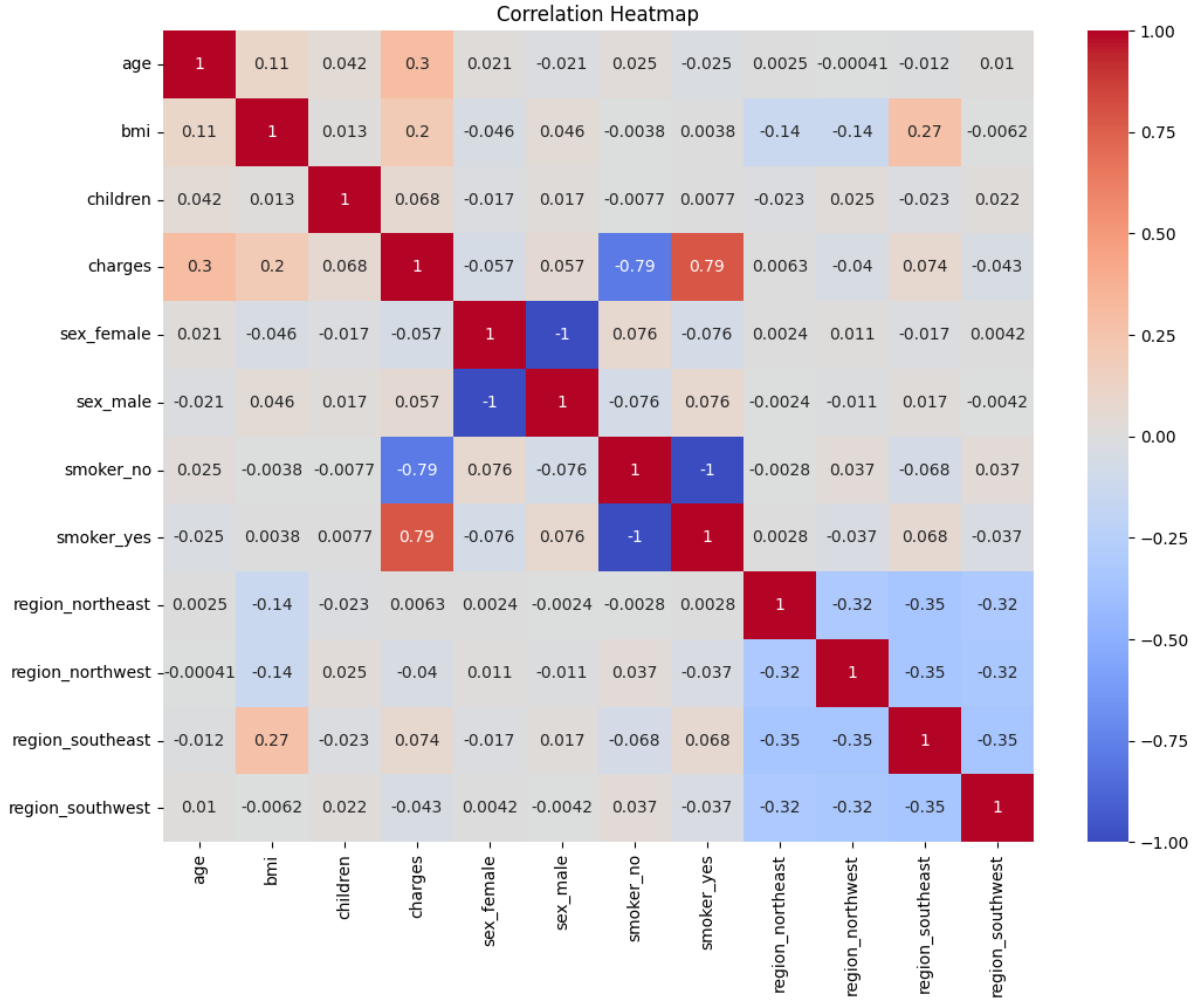


Before carrying on with the EDA I decided to do the Data Preprocessing in-between the EDA process as I wanted to include the categorical features in the **Correlation Heatmap** as it only takes numerical values. To solve this I used OneHotEncoder which converts categorical values into binary form. Once this was done, I carried on with the EDA.

Encode Categorical Variables (Data Preprocessing)

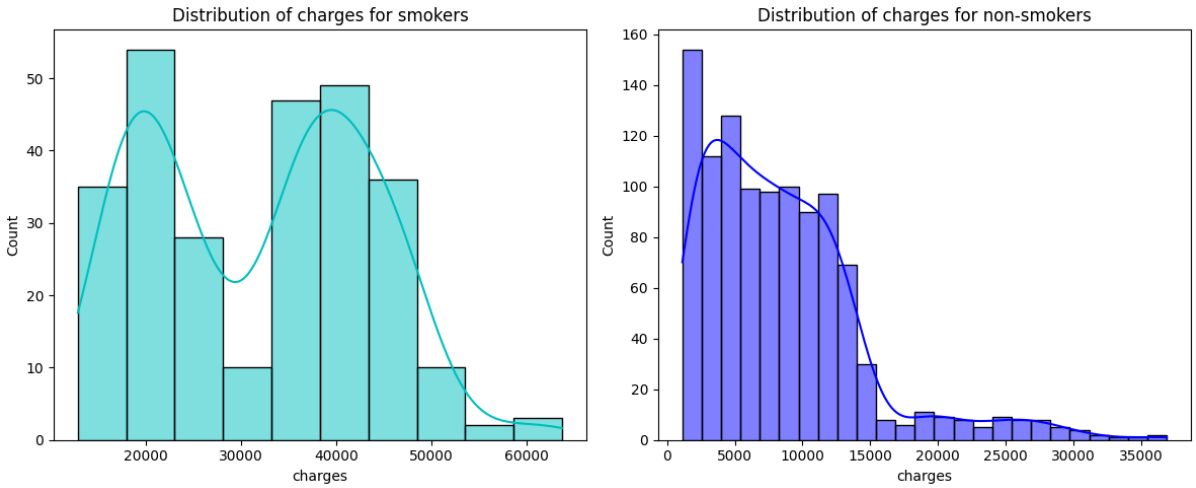
```
df2 = df.copy()

# Select categorical columns
cat_cols = df.select_dtypes('object').columns.tolist()

# Initialize and fit the OneHotEncoder
encoder = OneHotEncoder(sparse_output=False, handle_unknown='ignore')
encoder.fit(df2[cat_cols]) #(InsightsByRish, 2024)

# Transform categorical variables
encoded_cols = encoder.get_feature_names_out(cat_cols) #(InsightsByRish, 2024)
df2[encoded_cols] = encoder.transform(df2[cat_cols]) #(InsightsByRish, 2024)

# Drop original categorical columns
df2.drop(columns=cat_cols, inplace=True) #(InsightsByRish, 2024)
✓ 0.0s
```

- **Correlation Heatmap –** This is a chart which helps identify features that are strongly related to each other and to the target. This can also help guide with feature selection.



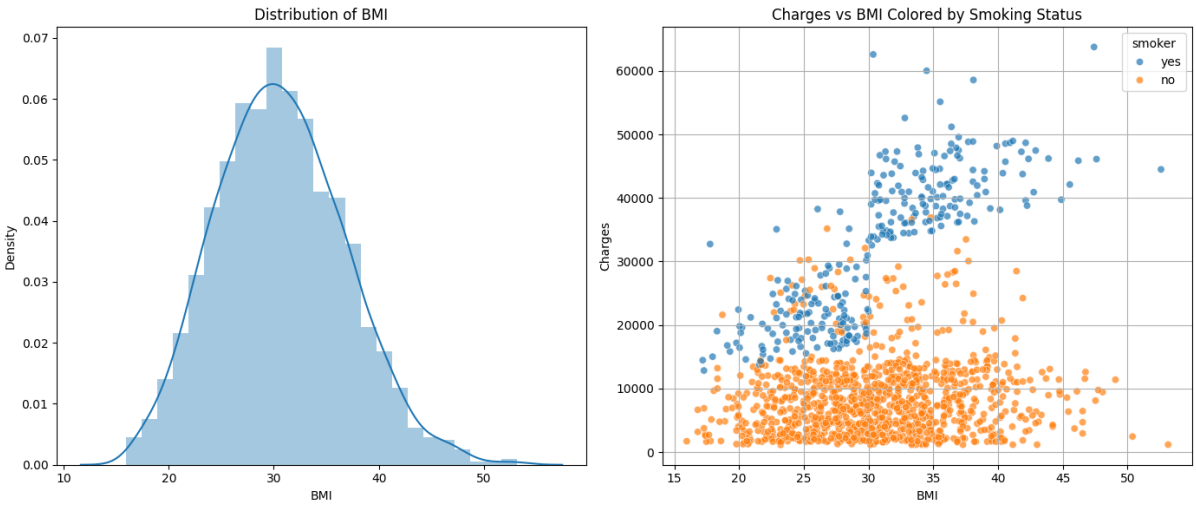
From this heatmap I was able to identify that those who smoke will pay more charges and that age and BMI does affect the charges too but not as significant.

- **Displot chart for smokers vs non smokers –** In this chart I wanted to see how charges was distributed across for those who smoke and those who don't smoke.

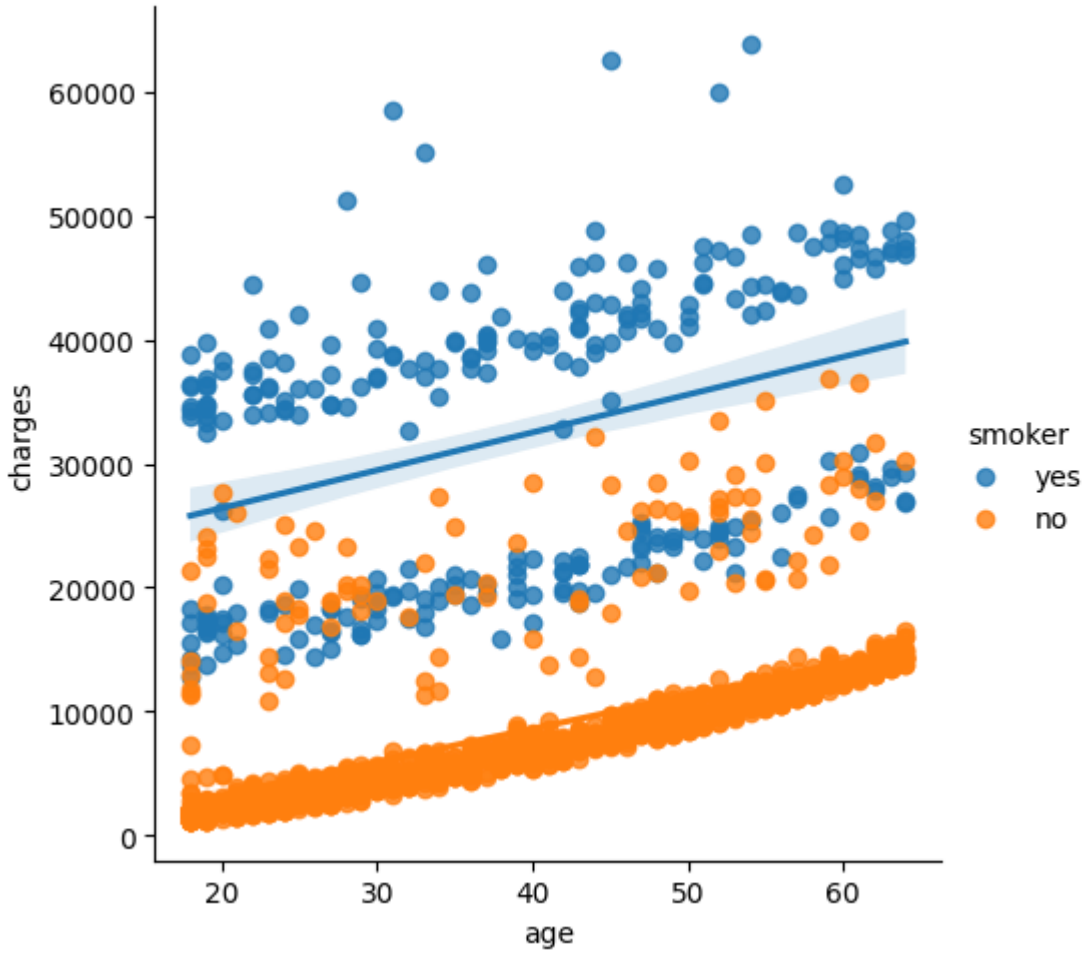


Based on the chart I was able to see that those who smoked paid a significantly higher amount than those who didn't smoke.

- **Displot and scatter plot for BMI and those who smoked –** The first plot shows the distribution of BMI and it showed that most of the patients had a BMI of around 30. The second plot shows that those who had a BMI of above 30 and smoked paid higher charges.



- **Lmplot for age vs charges by smoking status –** with this plot I wanted to see the relationship between peoples age of those who smoked and didn't against the charges they pay. In the end those who were older and smoked paid much higher charges.



Data Preprocessing

As mentioned earlier, I conducted the preprocessing early as I wanted to include some of the categorical values into the heatmap.

Defining features and targets

In order to help increase the models accuracy and performance, I defined 3 more features to include in the dataset. These features are:

- **BMI_smoker** – this feature is for those who have a BMI of above 30 and smoke
-

Feature Selection

The data is now ready to be used to train the model. Here I split the data into the features and target variables:

```
# Separate the independent variables (features) from the dependent variable (target)
X = df2.drop(['charges'], axis=1) # Features: everything except 'charges'
y = df2['charges'] # Target: the variable we are trying to predict
```

✓ 0.0s

The model

Train-Test Split

Once the features and target is separate, I split the data into a test set and a training set. This is to ensure that the model can get tested on data that it has not seen before. I used an 80 20 split where 80% of the data is used for training the model and the remaining 20% is used to test the model.

```
# Train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# 80% for training, 20% for testing; random_state ensures reproducibility
```

✓ 0.0s

Normalising the data

We normalise the data as it can help model performance and accuracy (Jaiswal, 2024). I used StandardScaler in this model.

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler() #(jonnybazookatone, 2017)
X_train = scaler.fit_transform(X_train)#(jonnybazookatone, 2017)
X_test = scaler.transform(X_test)#(jonnybazookatone, 2017)
```

Train the model

Finally we are ready to train the model. In this step I instantiate the model and then train the model using the training data.

```
# Initialize and train the Linear Regression model
model = LinearRegression() # Instantiate the model
model.fit(X_train, y_train) # Train the model using training data
```

✓ 0.0s

Make predictions

Once that was done, I then used the trained model to make predictions on the unseen test data.

```
# Use the trained model to make predictions on the test set
y_pred = model.predict(X_test)  # Predicts charges based on X_test features
```

✓ 0.0s

Evaluating the model

To evaluate if the model is a good model, I used sklearn libraries such as R-squared, mean absolute error(MAE), mean squared error (MSE) and root mean squared error (RMSE).

- R^2 Score: as it will show how well the model explains the variance in the target variable (Geeksforgeeks, 2023). A score closer to 1 means that the model predicted well whereas a score closer to 0 means that it performed poorly.
- Mean Absolute Error (MAE): as this tells us how much the model's predictions are off on average (Geeksforgeeks, 2023). A smaller value would be considered as good but it is important to note that this value is compared to the scale of the target variable.
- Mean Squared Error (MSE): as it calculates the average of the squared differences between actual and predicted values (Geeksforgeeks, 2023). It gives higher weight to larger errors, which means it penalises bigger mistakes more severely than MAE. A lower MSE indicates better model performance, but like MAE, it must be interpreted in the context of the scale of the target variable.
- Root Mean Squared Error (RMSE): RMSE is the square root of the mean squared error. It provides the error in the same units as the target variable (Geeksforgeeks, 2023). A lower RMSE means better fit. It is more sensitive to outliers than MAE.

Once the model was run through the metrics these are the results I got:

```python
# R² Score
r2 = r2_score(y_test, y_pred)

# Mean Absolute Error (MAE)
mae = mean_absolute_error(y_test, y_pred)

# Mean Squared Error (MSE)
mse = mean_squared_error(y_test, y_pred)

# Root Mean Squared Error (RMSE)
rmse = np.sqrt(mse)

# --- PRINT METRICS ---
print("Model Evaluation Results:")
print(f"R-squared (R²): {r2:.2f}")
print(f"Mean Absolute Error (MAE): {mae:.2f}")
print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")
```

✓ 0.0s

```
Model Evaluation Results:
R-squared (R²): 0.91
Mean Absolute Error (MAE): 2266.68
Mean Squared Error (MSE): 14924092.23
Root Mean Squared Error (RMSE): 3863.17
```

The model evaluation results show strong performance, with an R^2 score of 0.91, indicating that approximately 91% of the variance in medical charges is explained by the model. The MAE of 2266.68 and RMSE of 3863.17 suggest that, on average, the model's predictions deviate from the actual values by a few thousand dollars, which is reasonable given the wide range of charges in the dataset. The relatively low MSE further confirms that the model maintains consistent error rates without being heavily influenced by large individual errors.

To see if the models performance could be increased with regularization, I used the ridge regression library.

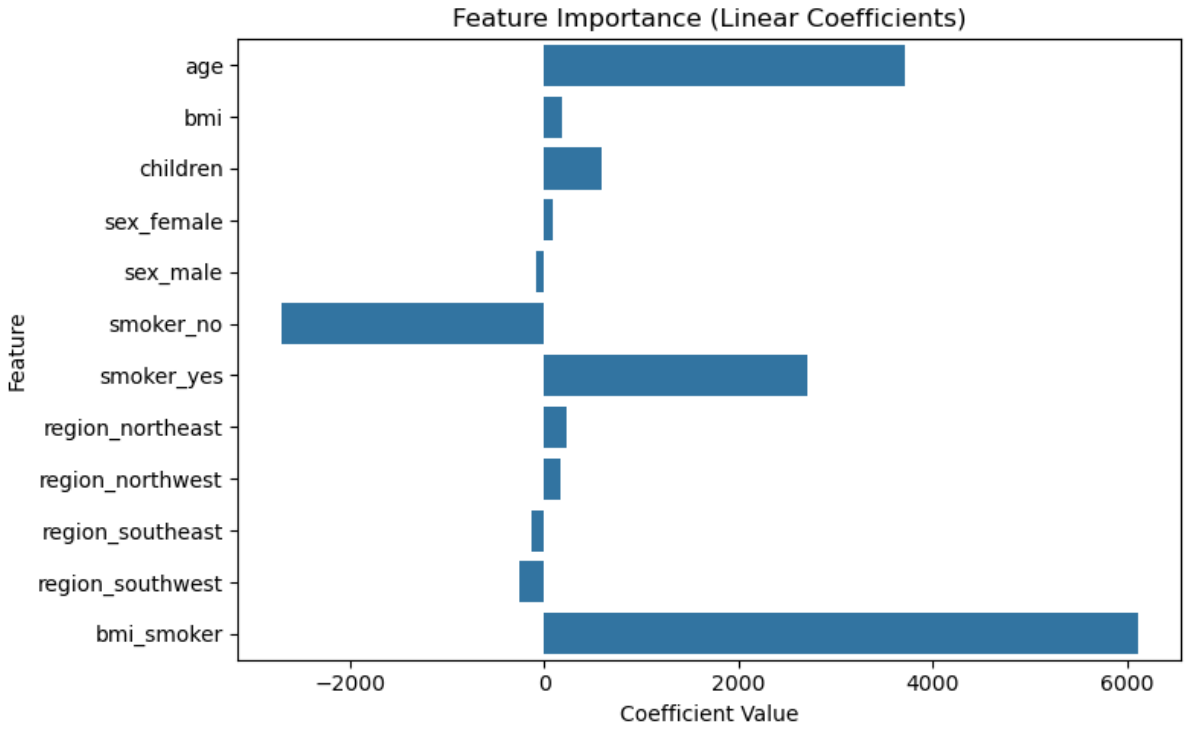
```
from sklearn.linear_model import Ridge
from sklearn.model_selection import cross_val_score

# Apply Ridge Regression with L2 regularization
ridge_model = Ridge(alpha=50, max_iter=100, tol=1.0)  #(codebasics, 2021)
ridge_model.fit(X_train, y_train)

# Evaluate with cross-validation

cv_scores = cross_val_score(ridge_model, X_test, y_test, cv=5, scoring='r2') #(scikit-learn, 2009) and (scikit-learn, n,d.)
print("Ridge Regression CV R² Scores:", cv_scores)
print("Average CV R² Score:", cv_scores.mean())
print(f"R² Score: {ridge_model.score(X_test,y_test):.2f}")
```

✓ 0.0s

```
Ridge Regression CV R² Scores: [0.85146853 0.94372117 0.90985607 0.77880681 0.93780022]
Average CV R² Score: 0.8843305595819496
R² Score: 0.91
```

From the results we can see that both the standard Linear Regression model and the Ridge Regression model with L2 regularization performed similarly, though the standard Linear Regression achieved the same R^2 score. The average cross-validated R^2 score for Ridge was 0.884, indicating that it generalises well across multiple data splits.

To show the linear coefficients and which coefficients effected the model the most I used a barplot as shown:

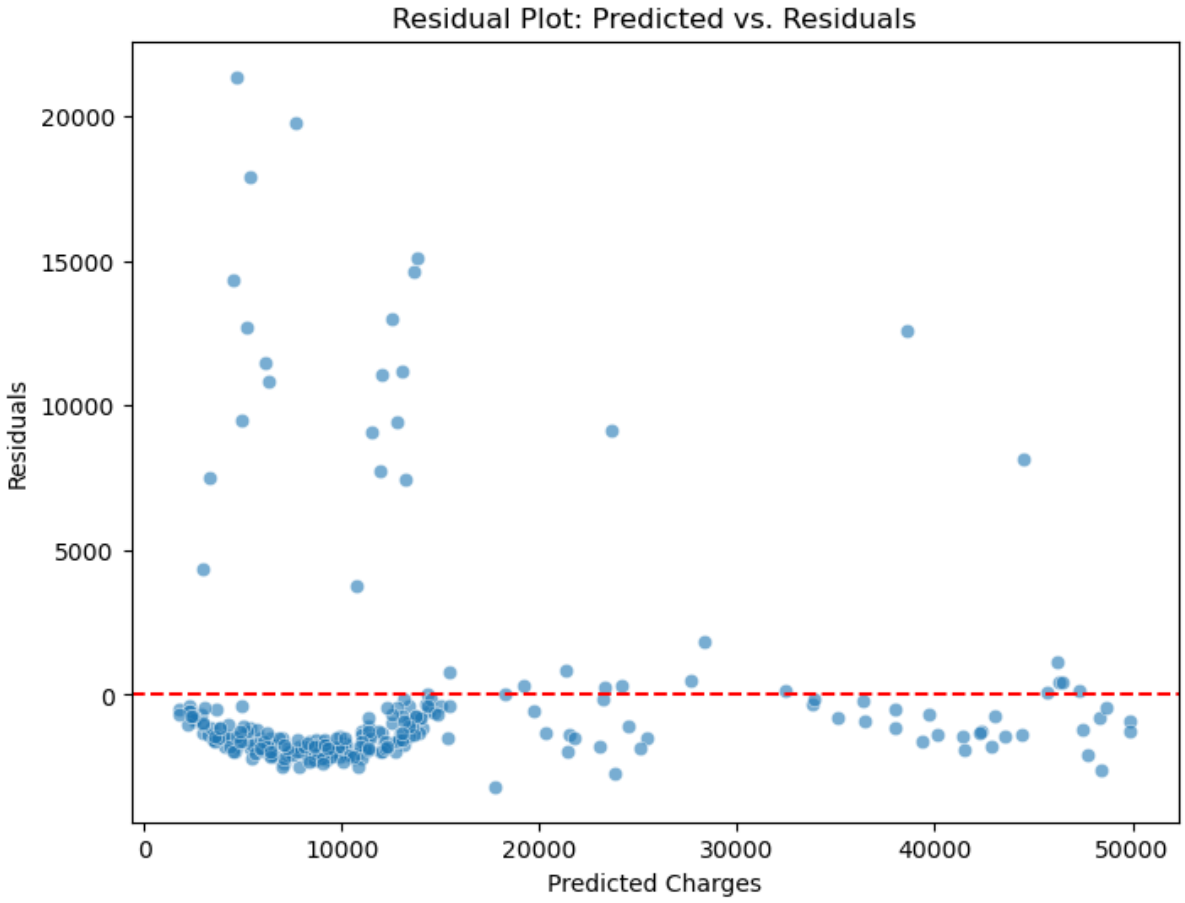


Here we can see that smoking and age as well as those who had a BMI of above 30 and smoked played a big role in the models performance.

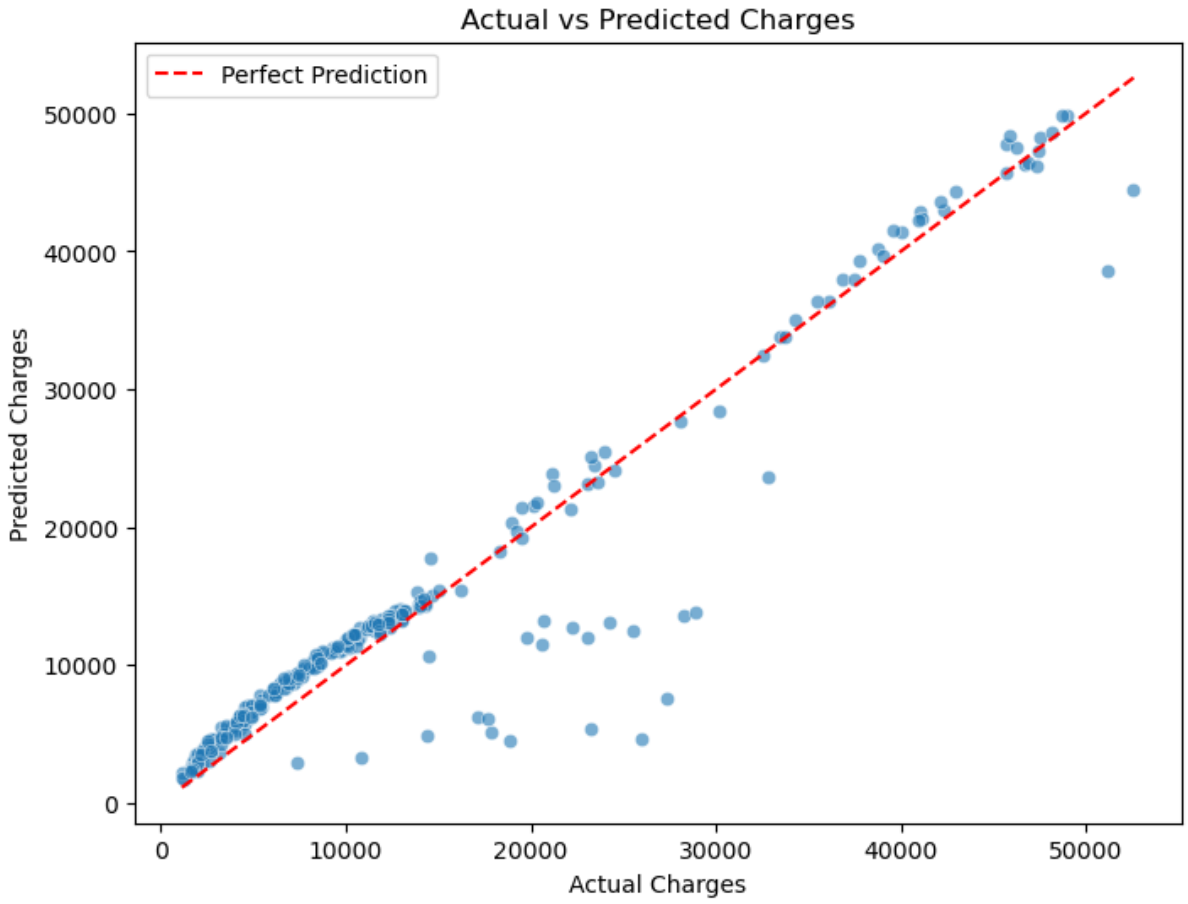
Visual Plots

I used two key graphs to visualise how well the model predicted:

- **Residual Plot**: This plot shows the difference between actual and predicted values (PennState, n.d.). Ideally, residuals should be randomly scattered around the horizontal axis (zero line), indicating a good fit. In the graph we can see that the plots are relatively close to the zero line with the exception of a few in the low and high ends. As depicted:



- **Actual vs Predicted Scatter Plot**: Ideally, all points should lie on the diagonal red line (perfect prediction). Our points were close for most cases with a few exceptions in the lower to middle area. However, majority of the points are relatively close together and are near the diagonal red line, indicating a good model. As shown:



Conclusion

In this project, a linear regression model was developed to predict medical charges based on demographic and lifestyle features such as age, sex, BMI, smoking status, number of children, and region. The dataset was carefully examined through exploratory data analysis, revealing that smoking status, age, and BMI are the most influential factors in determining medical charges.

The model achieved a high R^2 score of **0.91**, indicating that 91% of the variation in charges can be explained by the input features. The low MAE and RMSE values further confirm the model's strong predictive capability. Ridge Regression with L2 regularization and cross-validation was also applied to test for improvement. However, both models performed similarly, with standard linear regression slightly outperforming Ridge on the test set. This suggests that the model is already well-regularized and benefits from strong feature selection and low multicollinearity.

In conclusion, the model is both effective and interpretable, making it suitable as a proof-of-concept for medical aid cost prediction. Future work could explore non-linear models, additional features, or alternative datasets (e.g., local South African data) to further enhance prediction accuracy and real-world applicability.

Reference List

- GeeksforGeeks (2025). *ML | Linear Regression*. [Accessed 21 April 2025].
- GeeksforGeeks (2023). *Regression Metrics*. [Accessed 21 April 2025].
- PennState: Statistics Online Courses (n.d.). *Residual vs. Predictor Plot*. [Accessed 21 April 2025].
- GeeksforGeeks (2025). *What is Exploratory Data Analysis ?* [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/what-is-exploratory-data-analysis/. [Accessed 21 April 2025].
- Tableau (2024). *Data cleaning: the Benefits and Steps to Creating and Using Clean Data*. [online] Tableau Software. Available at: https://www.tableau.com/learn/articles/what-is-data-cleaning. [Accessed 25 April 2025].