

Planning

Project Overview

The goal of this project is to build a Linear Regression model to predict medical aid charges for clients, based on lifestyle and demographic features such as age, sex, BMI, smoking habits, number of children, and geographical region.

The dataset we'll use is based on medical charges in the United States, but we are using it as a proof-of-concept for a South African medical aid provider. The model will help assess whether dynamic pricing based on client characteristics can be feasibly implemented.

What Is Linear Regression?

According to Geeksforgeeks.org (2025), Linear regression is a supervised machine learning algorithm which uses labelled datasets to learn and create the best-fitting linear function for mapping data points. This function can then be used to make predictions on new data. The algorithm identifies the linear relationship between a dependent variable and one or more independent variables by fitting a linear equation to the observed data. Its primary goal is to predict continuous output values based on the input features. The image below depicts the equation of linear regression. It shows the dependent variable, intercept, coefficient and independent variable. Note that this equation only shows 1 independent variable with its coefficient as it is meant for a simple linear regression model. The dataset that we are working with is a multiple linear regression model as it contains more than 1 independent variable.

The diagram shows the linear regression equation $Y_i = \beta_0 + \beta_1 X_i$ with arrows pointing to each term from descriptive labels:

- Y_i is labeled "Dependent Variable" with an upward arrow.
- β_0 is labeled "Constant/Intercept" with a downward arrow.
- β_1 is labeled "Slope/Coefficient" with an upward arrow.
- X_i is labeled "Independent Variable" with a downward arrow.

This image was taken from (Meesala, 2023)

This image shows the difference in the equations between simple, multiple and polynomial models.

Simple Linear Regression	$y = b_0 + b_1x_1$
Multiple Linear Regression	$y = b_0 + b_1x_1 + b_2x_2 + \dots + b_nx_n$
Polynomial Linear Regression	$y = b_0 + b_1x_1 + b_2x_1^2 + \dots + b_nx_1^n$

This image was taken from (Tavishi, 2023)

Dataset Justification

The dataset used in this project, sourced from Kaggle (<https://www.kaggle.com/datasets/mirichoi0218/insurance>), is highly suitable for building a proof-of-concept linear regression model for predicting medical aid costs. Although the data is US-based, it contains relevant features such as age, sex, body mass index (BMI), number of children, smoking status, and geographic region—all of which are equally significant in the South African healthcare context. These variables influence healthcare costs universally and offer valuable insight into how different lifestyle and demographic factors correlate with insurance charges. The dataset is relatively clean, well-structured, and contains no missing values, making it ideal for initial modeling without requiring extensive preprocessing. Additionally, the target variable (charges) is continuous, aligning perfectly with the assumptions and requirements of linear regression. As a result, this dataset allows us to effectively explore relationships between variables and demonstrate how a sliding-scale pricing model could function, which supports the medical aid provider's goal of tailoring premiums to client profiles. Once validated, the same methodology can later be applied to locally sourced data for real-world deployment.

Exploratory Data Analysis (EDA) - Plan

EDA is the process of using visually and statistically understanding data and summarizing their main characteristics.

Step by Step Plan

1. Load the data

- In this step we load the dataset in via pandas so that it can be analysed and manipulated easily.
- The code we use is: `df = pd.read_csv("csvfilename.csv")`

2. Initial inspection of data and check for outliers

- In this step we get more insight with regards to the data.
- The code we use are:
 - `df.head()` shows a snapshot of the data for basic familiarisation.
 - `df.info()` helps check the data types and identify categorical vs numerical features.
 - `df.describe()` provides summary statistics such as mean, min, max, standard deviation, helping identify skewness, outliers, and range of features.

3. Check for missing values

- In this step we check if there are any missing values to ensure data quality as missing values can bias a model.
- The code we use is: `df.isnull().sum()`

4. Visualise numeric and categorical fields

- In this step we visualise the data in the form of graphs to help reveal data distributions, medians and outliers.
- We can understand the data more through the visualisations and can possibly help create more features

5. Check the correlation between the data

- Here we will check to see the relationship between the numeric features
- We will convert all categorical values into binary numbers so that they can be included in the correlation check using OneHotEncoder.
- The code we will use to check the correlation is:
 - `sns.heatmap(df.corr(), annot=True, cmap='coolwarm')`

Feature Selection

Based on the analysis done above we will determine what features we would like to use. Since this is a small dataset and each feature looks to have a logical impact on the medical charges we most probably will use all the features.

Model Training Plan

Split the data into training and testing sets

Once we have separated the features and the target value, we will then split the data into test and training data. We will use the value of 80% for training and 20% for testing.

- The code we are going to use is:
 - `X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)`

Create and Train the model

After splitting the data, we will then create and train the model.

We are using the `LinearRegression` class from `scikit-learn`, which fits a linear model using least squares. This step learns the optimal coefficients (β values) that minimize the prediction error (difference between predicted and actual charges).

- The code we are going to use is:
 - `model = LinearRegression()`
 - `model.fit(X_train, y_train)`

Model Evaluation

Make predictions

We will first make the predictions using the model.

- The code we will use is:
 - `y_pred = model.predict(X_test)`

Evaluate the model performance with metrics

Once the predictions are made we will then evaluate them against performance metrics.

The performance metrics we are going to use are:

- 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.

Visualise Actual vs Predicted values

To further evaluate our model we will compare the predicted vs actual values using a scatter plot graph. A good model will have the scatter dots close to the diagonal line. This will confirm whether the predicted charges are close to the actual charges and whether there is a consistent under or over estimation.

Report Planning

PDF Report

The final report will summarize the full analysis pipeline, and include visuals, code summaries, and interpretation of results. It will be structured clearly to communicate the work to both technical and non-technical stakeholders.

Report Structure

1. Introduction

- Problem statement
- Purpose of the model
- Brief overview of the dataset

2. Dataset Overview

- Source of the data
- Explanation of the features
- Target Variable

3. EDA Findings

- Summary of trends from plots
- Any outliers or skewness
- Any insights

4. Data Preprocessing and Model Training

- One-hot encoding
- Feature-target split
- Train-test split
- Description of linear regression

5. Model Evaluation

- MAE, MSE, RMSE, R^2 scores
- Visual plots

6. Conclusion

- Summary of findings
- Discussion of model limitations
- Potential improvements and next steps

Code Analysis

Importing Libraries

In [214...

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

import pandas as pd # Used for handling and manipulating tabular data (DataFrame o
import numpy as np # Provides support for mathematical operations and arrays
import matplotlib.pyplot as plt # Useful for creating basic data visualisations li
import seaborn as sns # Built on top of matplotlib; helps create more advanced, at
from sklearn.model_selection import train_test_split # Used to split the dataset i
from sklearn.linear_model import LinearRegression # Imports Linear Regression algo
```

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score, f1_
from sklearn.preprocessing import OneHotEncoder
```

Exploratory Data Analysis (EDA)

Loading Dataset (EDA)

This step is fundamental. We're inspecting data types, column names, and whether immediate cleaning is needed.

```
In [215... # Load the dataset into a Pandas DataFrame
df = pd.read_csv("insurance.csv") # Reads the CSV file into a structured table cal

# Display the first five rows of the dataset
df.head() # Useful for getting an initial understanding of the structure and conte
```

```
Out[215...
   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
```

Checking dataset structure (EDA)

```
In [216... df.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
```

Summary statistics (EDA)

```
In [217... # Get basic statistics about the dataset
df.describe() # Summarizes the distribution and spread of the numerical features
```

Out[217...

	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

We can see from the summary that there are some outliers in the BMI and a significant amount in the charges. In this model, we will not be removing the outliers — in the case of BMI, the outliers aren't extremely far from the norm, and in the case of charges, removing the outliers would eliminate valid and meaningful high-cost medical cases. These values are important for the model to learn how to predict the full range of real-world expenses accurately.

Check if any null values in the dataset (EDA)

Essential part of EDA: This ensures the dataset has no missing entries that might cause errors during model training.

In [218...

```
# Check for missing values in the dataset
df.isnull().sum() # This returns the count of null values per column; helps in ide
```

Out[218...

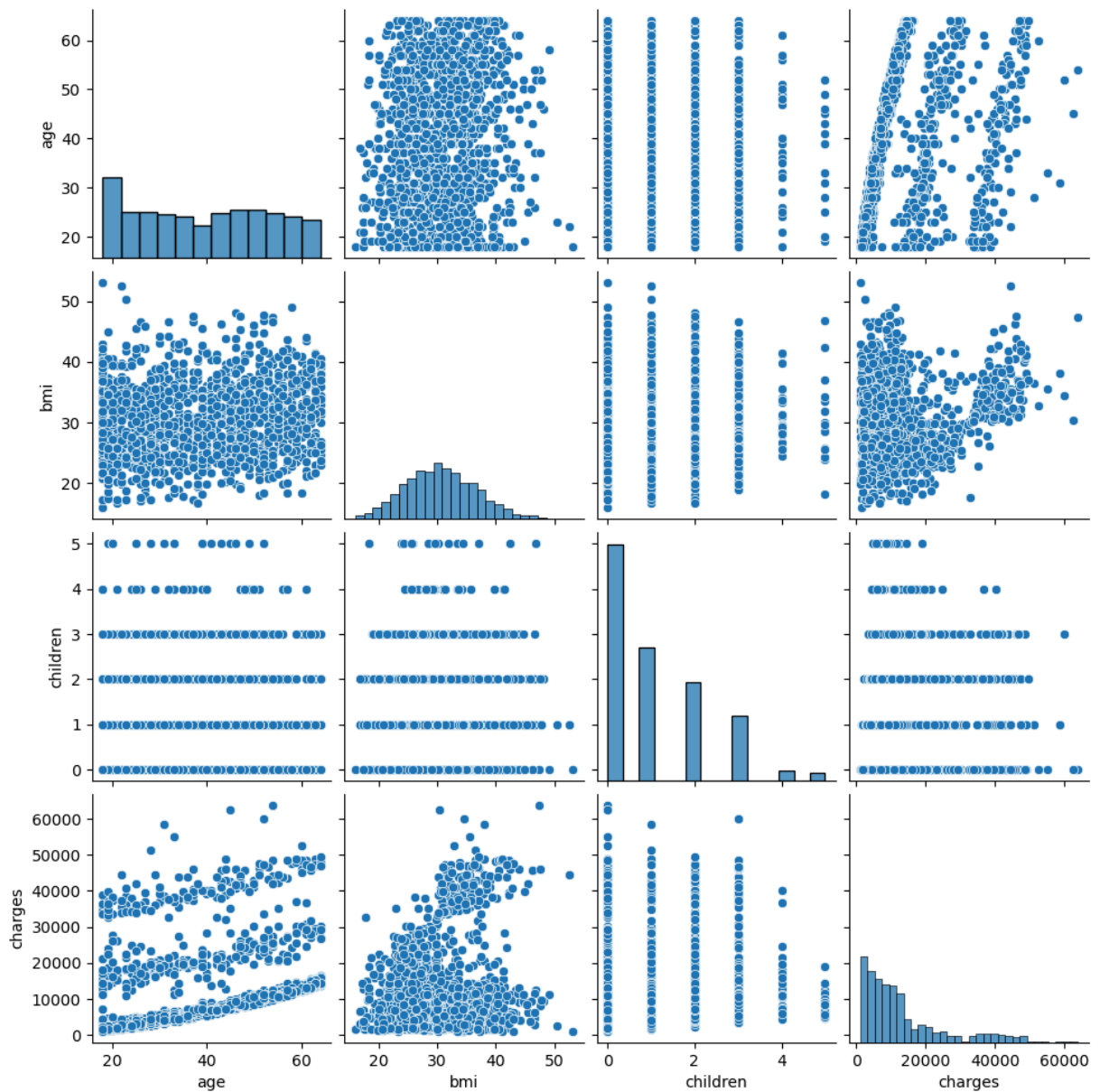
```
age      0
sex      0
bmi      0
children 0
smoker   0
region   0
charges  0
dtype: int64
```

Visualisation

Pairplot allows us to spot linear relationships, clusters, or patterns that might influence model accuracy.

In [219...

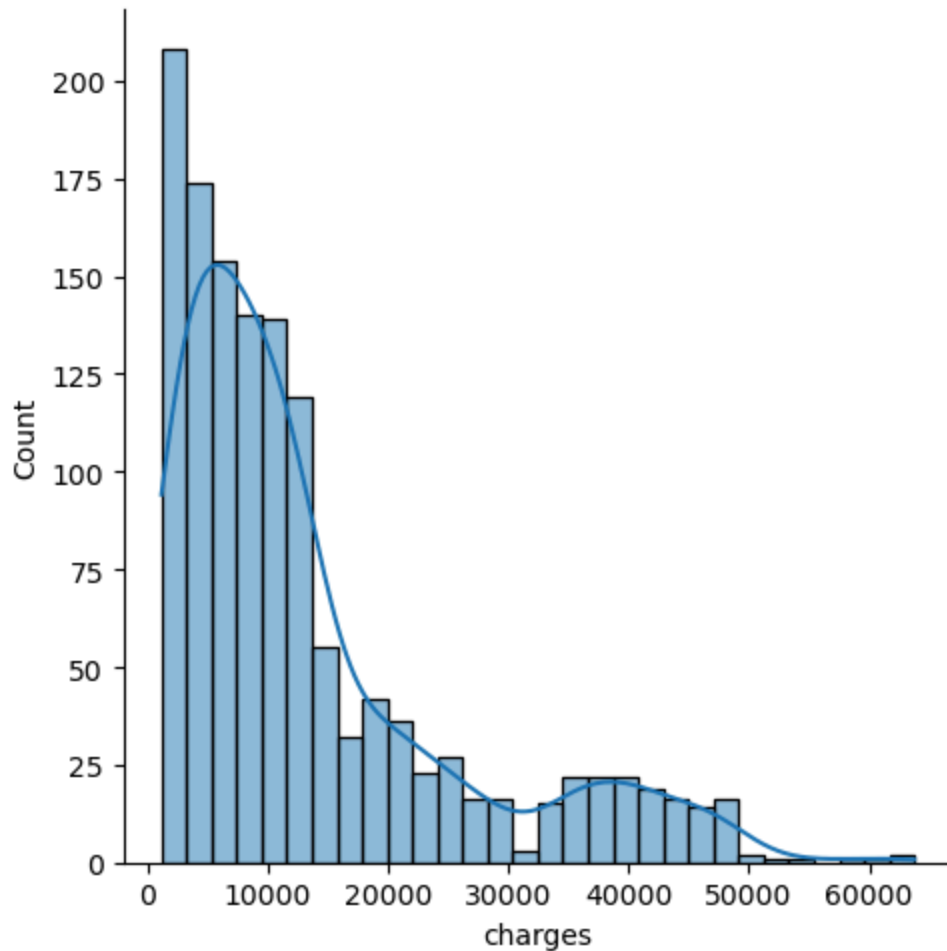
```
# Create pairplots to visually explore relationships between numerical features
sns.pairplot(df) # Displays scatter plots for each pair of variables and histogram
plt.show()
```

Distribution of charges (EDA)

```
In [220...] sns.displot(df['charges'], kde=True)
```

```
Out[220...] <seaborn.axisgrid.FacetGrid at 0x23ec4c3c9e0>
```



Encode Categorical Variables (Data Preprocessing)

In [221...

```
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)

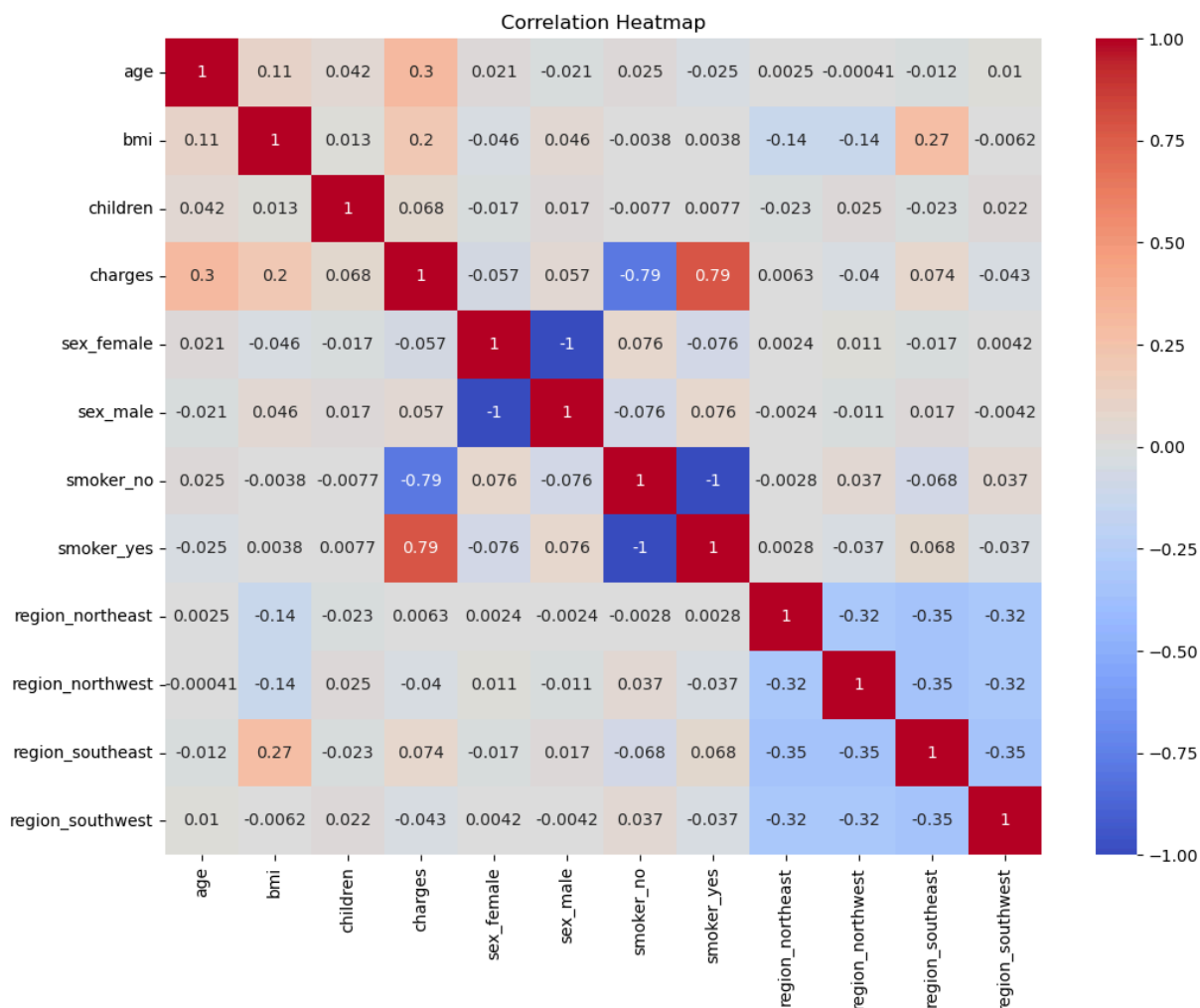
# Drop original categorical columns
df2.drop(columns=cat_cols, inplace=True) #(InsightsByRish, 2024)
```

Correlation heatmap (EDA)

Identifies features that are strongly linearly related to charges (target).

Helps guide feature selection decisions.

```
In [222... # Generate a correlation matrix to identify highly correlated variables
plt.figure(figsize=(12, 9))
sns.heatmap(df2.corr(), annot=True, cmap="coolwarm") # Display correlation coefficient
plt.title("Correlation Heatmap")
plt.show()
```



From the above correlation matrix we can see that those who smoke will pay more charges. We can also play somewhat of a role with the charges

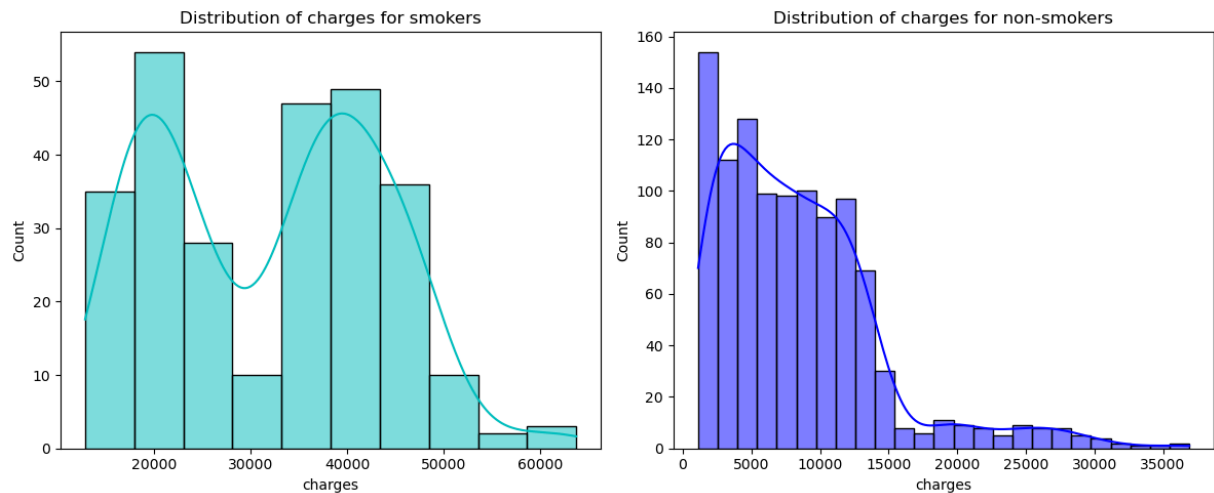
Checking the distribution of charges for smokers vs non smokers (EDA)

```
In [223... plt.figure(figsize=(12, 5))

# Plot for smokers
plt.subplot(1, 2, 1)
sns.histplot(df2[df2['smoker_yes'] == 1]['charges'], color='c', kde=True)
plt.title('Distribution of charges for smokers')

# Plot for non-smokers
plt.subplot(1, 2, 2)
sns.histplot(df2[df2['smoker_no'] == 1]['charges'], color='b', kde=True)
plt.title('Distribution of charges for non-smokers')
```

```
plt.tight_layout()
plt.show()
```



From this chart we can derive that those who smoke pay higher charges

BMI vs Charges (EDA)

In [224...

```
import matplotlib.pyplot as plt
import seaborn as sns

# First graph: Distribution of BMI
plt.figure(figsize=(14, 6))

plt.subplot(1, 2, 1)
sns.distplot(df["bmi"])
plt.title("Distribution of BMI")
plt.xlabel("BMI")
plt.ylabel("Density")

# Second graph: Scatterplot of Charges vs BMI colored by Smoking Status
plt.subplot(1, 2, 2)
sns.scatterplot(data=df, x='bmi', y='charges', hue='smoker', alpha=0.7)
plt.title("Charges vs BMI Colored by Smoking Status")
plt.xlabel("BMI")
plt.ylabel("Charges")
plt.grid(True)

plt.tight_layout()
plt.show()
```

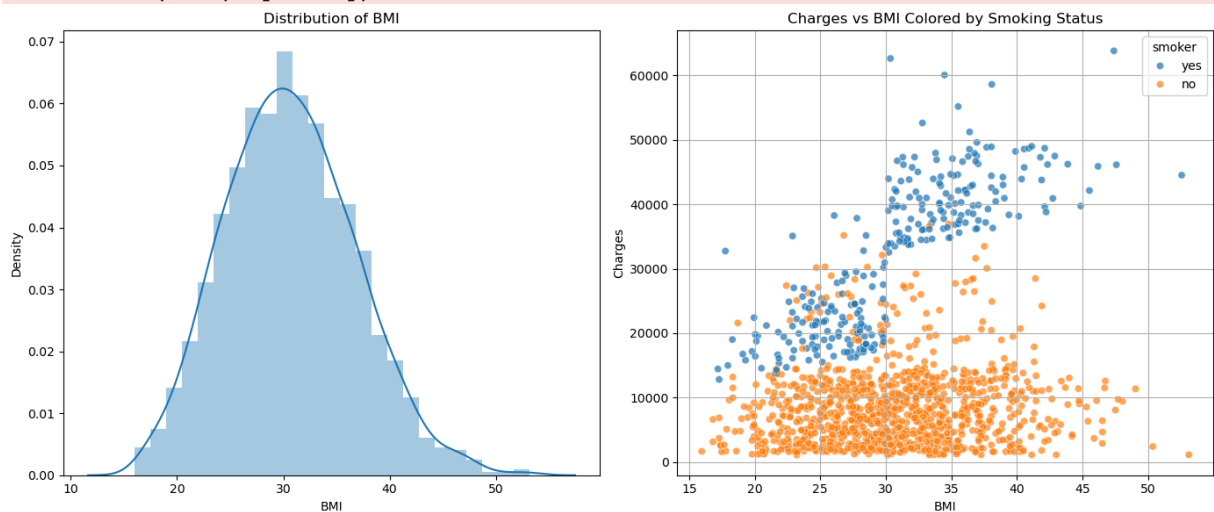
C:\Users\lab_services_student\AppData\Local\Temp\ipykernel_7028\985105318.py:8: User Warning:

``distplot`` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either ``displot`` (a figure-level function with similar flexibility) or ``histplot`` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see <https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751>

```
sns.distplot(df["bmi"])
```

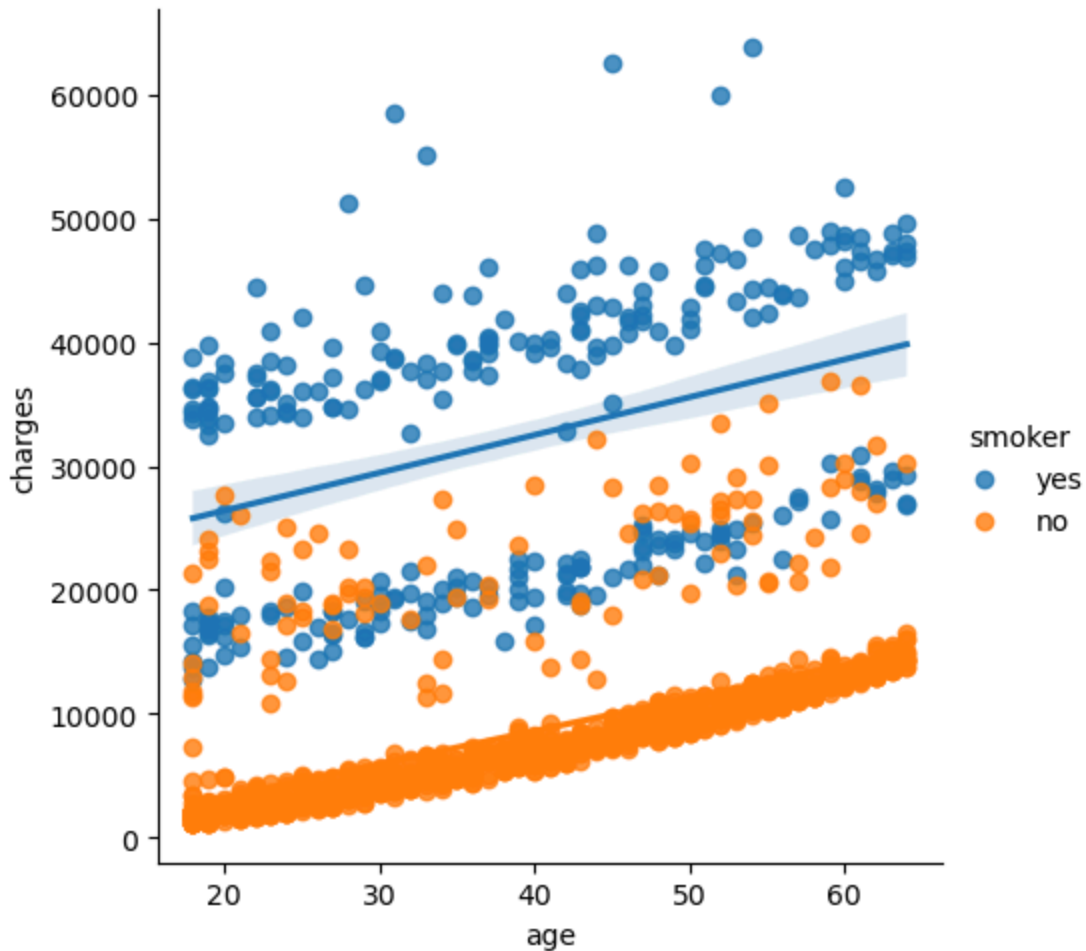


From this chart we can derive that those who have a BMI of above 30 and smoke will pay more charges

Charges distribution for Age Value and Smokers (EDA)

```
In [225... sns.lmplot(x="age", y="charges", hue="smoker", data=df)
```

```
Out[225... <seaborn.axisgrid.FacetGrid at 0x23ec17a6930>
```



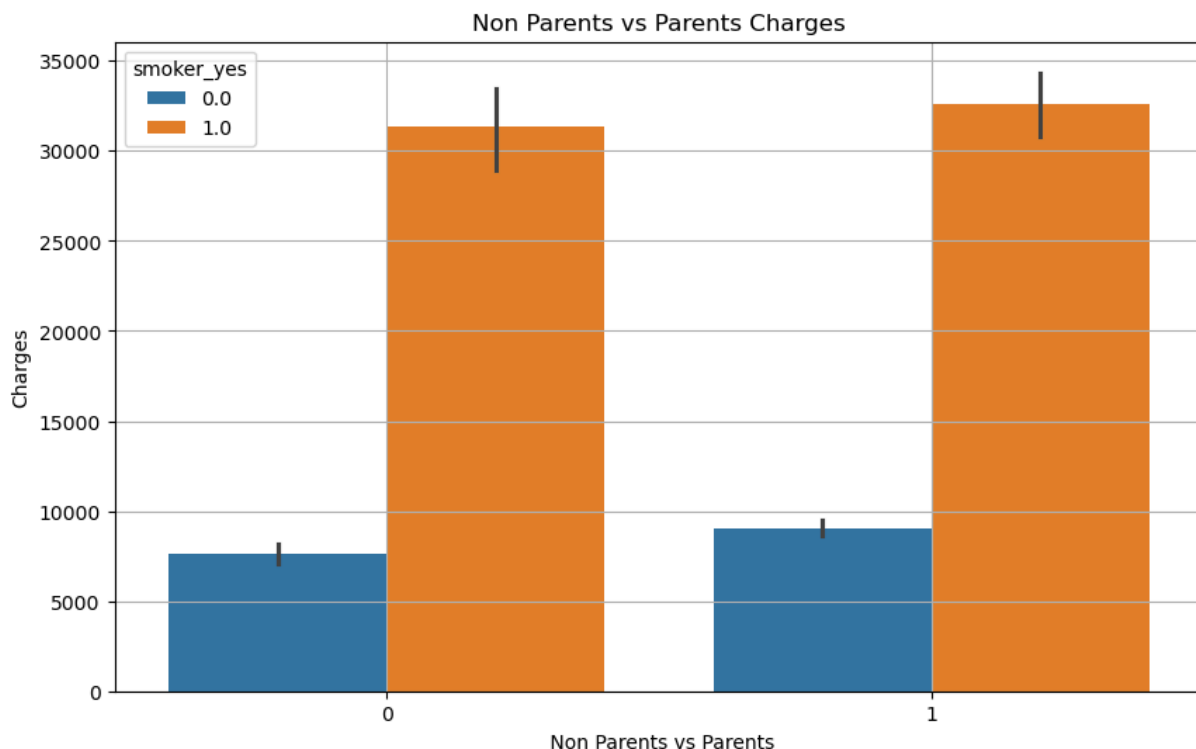
From this graph we can see that as age increases, so does the cost of charges. We can also see that if you smoke you have a much higher chance to pay more charges

Define Features & Target (Data Preprocessing)

Adding additional features

```
In [226... df2['bmi_smoker'] = df2['bmi'] * df2['smoker_yes'] #based on the graph created above
df2['is_parent'] = df2['children'].apply(lambda x: 1 if x > 0 else 0) #typically if
df2['aged_smoker'] = df2['age'] * df2['smoker_yes']
```

```
In [227... #This graph shows the difference in charges Parents vs Non Parents pay
plt.figure(figsize=(10, 6))
sns.barplot(data=df2, x='is_parent', y='charges', hue='smoker_yes')
plt.title("Non Parents vs Parents Charges")
plt.xlabel(" Non Parents vs Parents")
plt.ylabel("Charges")
plt.grid(True)
plt.show()
```



From this graph we can derive that if someone is a parent they will pay a bit more than someone who isn't but it is not a significant amount. We will still use this in the model however as in the future being a parent might affect the cost significantly.

Exporting updated dataset to new CSV

In [228... `df2.to_csv("Insurance New") #(Pdeuxa, 2020)`

We export the cleaned and updated dataset into a new CSV for the event of using it in future newer models.

Feature Selection for training.

In [229... `# 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`

note that we don't drop both BMI and Smoker columns as it would reduce model performance, because you lose the base-level effect of smoking or BMI independently.

Train-Test Split

Prevents overfitting and enables fair model evaluation.

In [230... `# Train/test split`
`X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta`

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

Train the Model

According to Geeksforgeeks.org (2025), Linear regression is a supervised machine learning algorithm which uses labelled datasets to learn and create the best-fitting linear function for mapping data points. This function can then be used to make predictions on new data.

```
In [231... # Initialize and train the Linear Regression model
model = LinearRegression() # Instantiate the model
model.fit(X_train, y_train) # Train the model using training data
```

```
Out[231... LinearRegression
LinearRegression()
```

```
In [232... model.coef_ #independent values
```

```
Out[232... array([ 2.63750633e+02,  1.82106154e+01,  4.57432053e+02,  2.18350091e+02,
        -2.18350091e+02,  1.00197933e+04, -1.00197933e+04,  6.94849804e+02,
         1.82715412e+02, -3.87818443e+02, -4.89746773e+02,  1.42696722e+03,
         1.95545698e+02,  1.55249722e+00])
```

```
In [233... model.intercept_ #constant value
```

```
Out[233... -13084.06009597552
```

Make predictions

This step tests the model's generalization to unseen data.

```
In [234... # Use the trained model to make predictions on the test set
y_pred = model.predict(X_test) # Predicts charges based on X_test features
```

Evaluating the model

These metrics give a comprehensive view of model accuracy and error spread.

We use the following:

- 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.

In [235...

```
# 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}")
```

Model Evaluation Results:
 R-squared (R²): 0.88
 Mean Absolute Error (MAE): 2858.25
 Mean Squared Error (MSE): 19196540.55
 Root Mean Squared Error (RMSE): 4381.39

Evaluation Metrics Summary

The model evaluation results show strong performance, with an R² score of 0.88, indicating that approximately 88% of the variance in medical charges is explained by the model. The MAE of 2858.25 and RMSE of 4381.39 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.

Evaluation Visualisation

Here we use a Residual vs Predicted Plot Graph and a Actual vs Predicted Scatter Plot graph.

Residual vs Predicted: A residual vs. predictor plot is a scatterplot that displays the residuals (the differences between observed and predicted values) on the y-axis and the predictor (independent variable) on the x-axis (PennState: Statistics Online Courses, n.d.). We calculate residuals by taking the actual values and subtracting it from the predicted values.

Residuals = actual - predicted

We use this to check linearity. If the residuals show a random scatter around zero, it suggests a linear relationship is appropriate. Patterns like curves or trends indicate the model might be missing non-linear effects.

Actual vs Predicted:

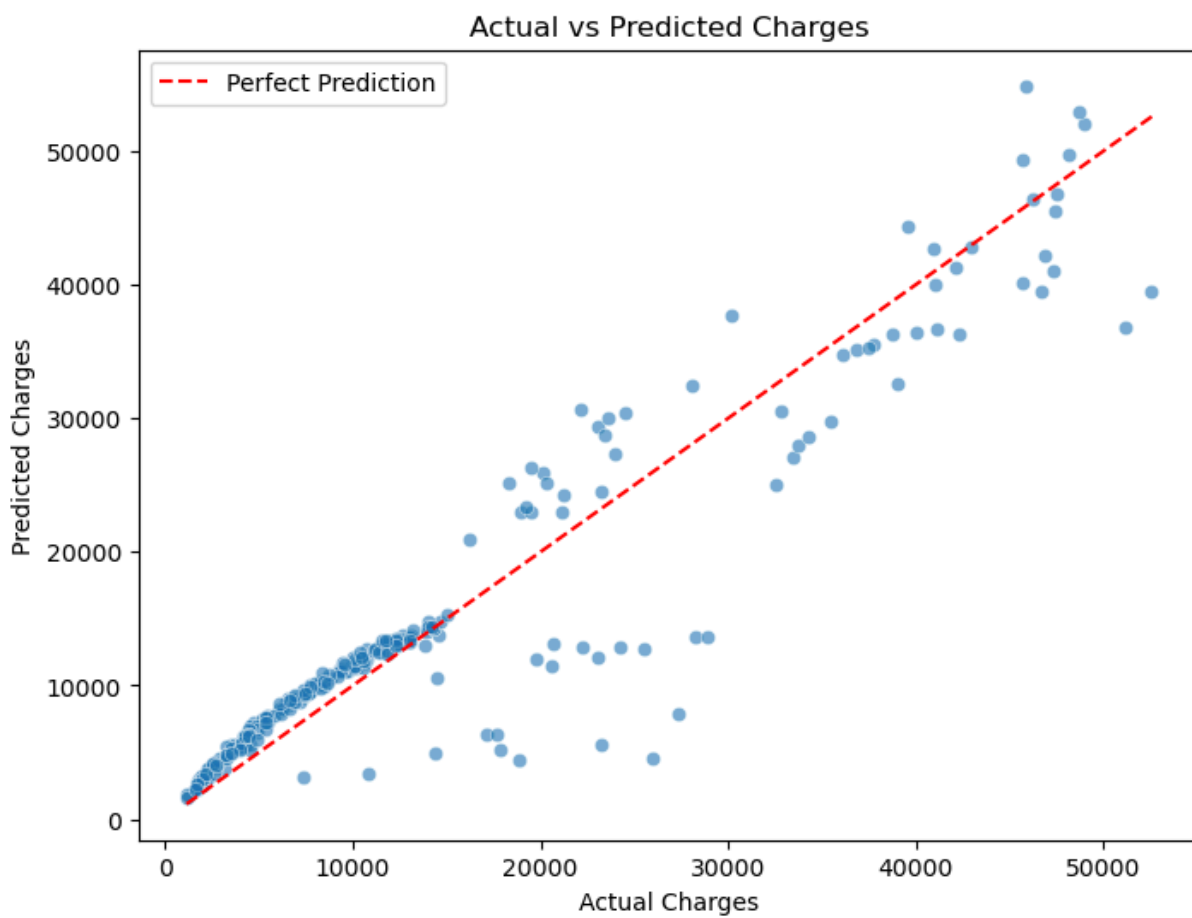
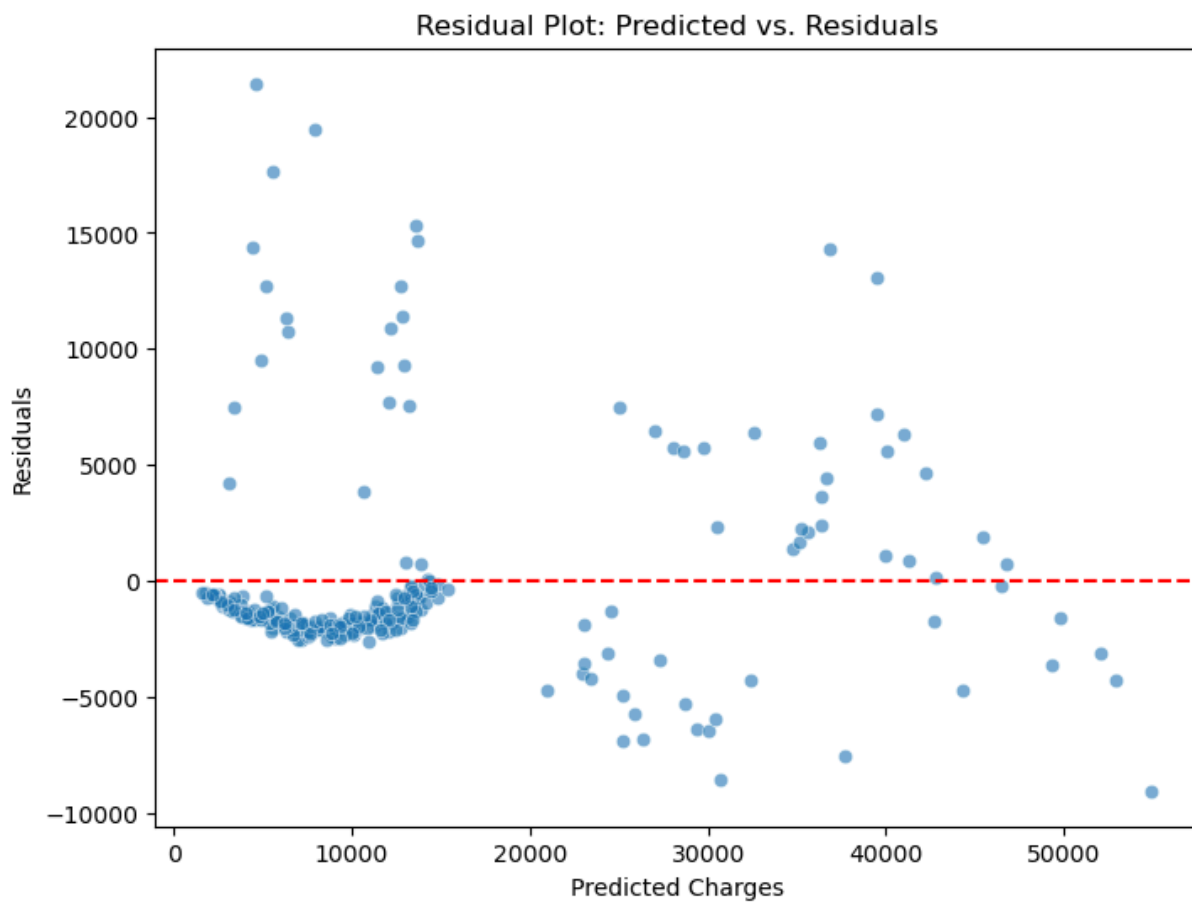
In [236...

```
# VISUALISATION: RESIDUAL PLOT

residuals = y_test - y_pred #(PennState: Statistics Online Courses, n,d.)

plt.figure(figsize=(8, 6))
sns.scatterplot(x=y_pred, y=residuals, alpha=0.6)
plt.axhline(0, color='red', linestyle='--')
plt.title('Residual Plot: Predicted vs. Residuals')
plt.xlabel('Predicted Charges')
plt.ylabel('Residuals')
plt.show()

# VISUALISATION: ACTUAL VS PREDICTED
plt.figure(figsize=(8, 6))
sns.scatterplot(x=y_test, y=y_pred, alpha=0.6)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], '--r', label='y=x')
plt.title('Actual vs Predicted Charges')
plt.xlabel('Actual Charges')
plt.ylabel('Predicted Charges')
plt.legend()
plt.show()
```



Interpretation of the graphs

Residual vs. Predicted Plot:

This plot displays the residuals (the difference between actual and predicted values) against the predicted charges. Ideally, residuals should be randomly scattered around the horizontal axis (zero line), indicating a good fit (PennState: Statistics Online Courses, n.d.). However, the plot shows a clear curved (U-shaped) pattern, suggesting a non-linear relationship that the model fails to capture. Additionally, the spread of residuals increases with higher predicted charges, indicating heteroscedasticity—where the variance of errors is not constant. These signs suggest that the current model may be too simple and is not adequately capturing the complexity of the data.

Actual vs. Predicted Plot:

This scatter plot compares the actual charges to the model's predicted charges, with the red dashed line representing perfect predictions. While the model performs well for lower and mid-range charges (as points are close to the line), it consistently underestimates higher charges, with many predictions falling below the perfect prediction line. This trend indicates that the model lacks accuracy at the upper end of the target variable range and may be missing important relationships or features.

Conclusion

Overall, this analysis offers valuable insights into the drivers of medical expenses and provides a strong foundation for building predictive tools in the insurance and healthcare sectors. With further improvements such as trying regularization techniques or more complex models, the predictive power could be enhanced even further.

Reference List

Geeksforgeeks.org. (2025). ML | Linear Regression - GeeksforGeeks. [online] GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/ml-linear-regression/>. [Accessed 21 April 2025].

Geeksforgeeks.org. (2021). ML | Python – seaborn.Implot() method. [online] GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/python-seaborn-Implot-method/> [Accessed 21 April 2025].

GeeksforGeeks. (2023). Regression Metrics. [online] Available at: <https://www.geeksforgeeks.org/regression-metrics/>. [Accessed 21 April 2025].

InsightsByRish (2024). House Price Prediction Project | Linear Regression | Machine Learning. [online] YouTube. Available at: https://www.youtube.com/watch?v=x_GZgeLJU4g [Accessed 21 April 2025].

Léonard. (2020). How to show min and max values at the end of the axes. [online] Stack Overflow. Available at: <https://stackoverflow.com/questions/64321561/how-to-show-min-and-max-values-at-the-end-of-the-axes>. [Accessed 21 April 2025].

matplotlib.org. (n.d.). Choosing Colormaps in Matplotlib — Matplotlib 3.8.0 documentation. [online] Available at: <https://matplotlib.org/stable/users/explain/colors/colormaps.html>. [Accessed 22 April 2025].

Meesala, P., 2023. A Brief Overview of Simple Linear Regression and its Implementation in Python. [online] Medium. Available at: <https://prasad07143.medium.com/a-brief-overview-of-simple-linear-regression-and-its-implementation-in-python-e6b5c180ff08> [Accessed 21 April 2025].

Pdeuxa (2020). Saving my dataset as a csv file in python. [online] Stack Overflow. Available at: <https://stackoverflow.com/questions/62396245/saving-my-dataset-as-a-csv-file-in-python>.

PennState: Statistics Online Courses. (n.d.). 4.3 - Residuals vs. Predictor Plot | STAT 501. [online] Available at: <https://online.stat.psu.edu/stat501/lesson/4/4.3>. [Accessed 21 April 2025].

Tavishi, 2023. Linear Regression in ML. [online] Medium. Available at: <https://medium.com/@tavishi.1402/linear-regression-in-ml-56107a59be03> [Accessed 21 April 2025].