ST1 Capstone Programming Project

This project will be analysing the attributes of a medical insurance dataset to create a model for prediction of incurred charges. [**GITHUB LINK**](https://github.com/skullermoji/ST1_Capstone)

My approach to this task will be incorporating the Python libraries **tkinter**, **pandas**, **seaborn**, and **matplotlib**.

Therefore, before we move on we shall import these libraries:

import tkinter as tk

from tkinter import filedialog, messagebox

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

**Step 1.**

*Reading the Dataset*

# Load dataset

data = pd.read\_csv('medical\_insurance.csv')

# Display first few rows of the dataset in consoel

print(data.head())

# Display general information and basic statistics in console

print(data.info())

print(data.describe())

A screenshot of a computer screen

Description automatically generatedA screenshot of a computer code

Description automatically generatedA close up of words

Description automatically generated

**Key observations from Step 1**

* The dataset is a sample of 2772 people with medical insurance
* The attributes include age, bmi, number of children, charges, gender, smoker, and region.
* Gender, smoker and region attributes are functionally Booleans, being True or False. For example, sex\_male = False would indicate a female.

**Step 2.**

*Problem Statement Definition*

* Creating a prediction model in order to predict the price of medical insurance based on given attributes.

**Step 3.**

*Target Variable Identification*

* Target Variable: Charges (Affected by: Age, Sex, Smoker, Region, etc)

**Step 4.**

*Choosing appropriate Algorithm for Data Analysis*

* The target variable is Continuous, and we shall be proceeding with a Ridge regression model as we expect a linear relationship.

*Visualising the distribution of Target Variable*

# Pair plot to examine relationships among numerical features

sns.pairplot(data)

A graph of a number of people

Description automatically generated with medium confidenceplt.show()

* We can see that the data has a relatively linear relationship, with most charges being weighted towards the lower end.
* We shall proceed and investigate how the other variables affect this.

**Step 5.**

*Data Exploration at the Basic Level*

* As per Step 1.

# Display first few rows of the dataset in consoel

print(data.head())

# Display general information and basic statistics in console

print(data.info())

A close up of words

Description automatically generatedA screenshot of a computer code

Description automatically generatedA screenshot of a computer screen

Description automatically generatedprint(data.describe())

* We expect that the target variable will be affected by number of *children*, *sex*, whether they are a *smoker*, and their *region*. BMI could be a potential factor.
* Age – Continuous
* BMI – Continuous
* Children – Continuous
* Charges – Continuous
* Sex – Categorical
* Smoker – Categorical
* Region – Categorical

**Step 6.**

*Identifying and Rejecting unwanted columns*

* Given our variables, there should be no need to remove any columns
* I suspect they shall all have some effect

**Step 7.**

*Visual Exploratory Data Analysis*

# Pair plot to examine relationships among numerical features

sns.pairplot(data)

A collage of graphs

Description automatically generatedplt.show()

A chart of different sizes and colors

Description automatically generated with medium confidenceA diagram of a chart

Description automatically generated with medium confidenceA chart of a person and person

Description automatically generated

**Observations from Step 7.**

* The pairplot shows the non-categorical data from the dataset.
* The variables are represented on both X and Y axis
* We can see how some variables such as BMI form a nice bell curve indicating a sufficient and diverse sample of people
* Some charts are less relevant although still do have features that indicate a relationship.
* As for the bar charts, I noticed by far the most difference between whether a person smoked or not. If they did, they had to pay a substantial amount more than a non-smoker, which makes sense.
* Males cost more on average along with greater standard deviation
* Some regions have larger standard deviations and extremes although all had a similar median

**Step 8.**

*Feature Selection based on data distribution*

* Despite some graphs being rather dense due to the dataset density, all show signs of being relevant to the target variable.
* Therefore we shall move on and analyse further

**Step 9.**

*Outliers and Missing Values*

# Calculates the Interquartile Range for charges

Q1 = data['charges'].quantile(0.25)

Q3 = data['charges'].quantile(0.75)

IQR = Q3 - Q1

# Define bounds

lower\_bound = Q1 - 1.5 \* IQR

upper\_bound = Q3 + 1.5 \* IQR

# Filter the dataset to remove outliers

filtered\_data = data[(data['charges'] >= lower\_bound) & (data['charges'] <= upper\_bound)]

# Check the size of the filtered dataset compared to the original dataset

print("Original dataset size:", len(data))

print("Filtered dataset size:", len(filtered\_data))

* After running this code, we have trimmed down the dataset from 2772 to 2476, which is a moderate amount of “outliers” given the total and suggests that the data is relatively consistent.

**Step 10.**

*Visualising data after outlier removal*

# Pair plot to examine relationships among numerical features

sns.pairplot(filtered\_data)

plt.show()

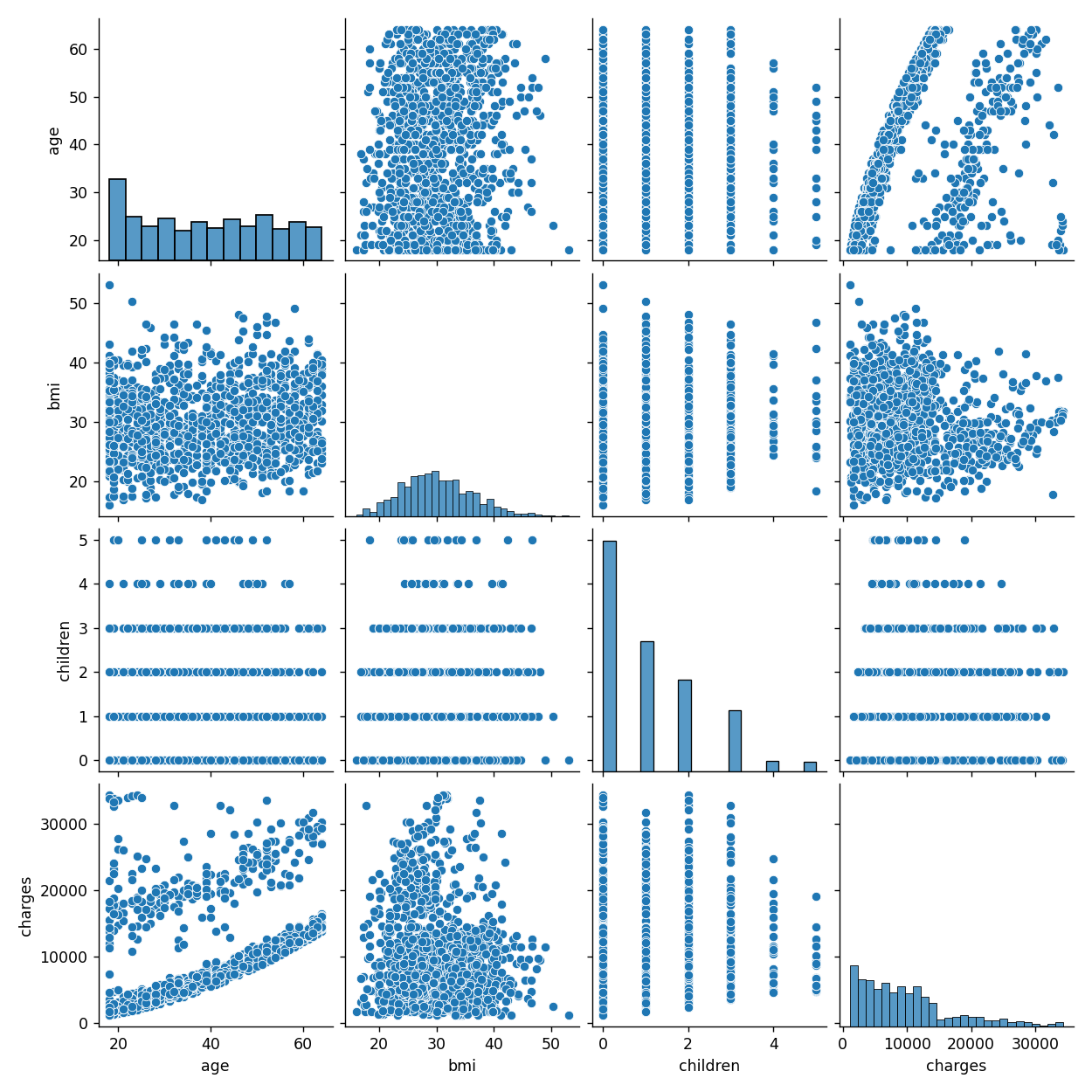
# Analyse categorical variables using boxplots

for column in ['sex', 'smoker', 'region']:

    sns.boxplot(x=column, y='charges', data=filtered\_data)

    plt.title(f"Charges vs {column}")

    plt.show()

* Data has been changed to filtered data for analysis

**Observations from Step 10.**

* There is little visible difference in the pairplot although certainly present
* Greatest change can be seen in the target variable ‘charges’
* The Interquartile Range filtering may have adverse effects on the overall data as now it potentially may be too consistent
* We no longer have charges approaching 20K
* Tail is still fairly linear

**Step 11.**

*Data Conversion to numeric values for machine learning*

# Convert categorical variables to one-hot encoded format

data = pd.get\_dummies(data, columns=['sex', 'smoker', 'region'], drop\_first=True)

* Convert categorical data into pandas dummy variables
* drop\_first=True to filter out ‘Unknown’ gender
* Picked these variables as they seemed to be the best features

**Step 12.**

*Training/Testing*

from sklearn.model\_selection import train\_test\_split

# Define features and target variable

X = data.drop('charges', axis=1)

y = data['charges']

# Split into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

from sklearn.linear\_model import Ridge

from sklearn.metrics import mean\_squared\_error, r2\_score

# Create and train the Ridge regression model

model = Ridge(alpha=1.0) # Default alpha 1.0 as per Ridge documentation

model.fit(X\_train, y\_train)

# Make predictions on the test set

y\_pred = model.predict(X\_test)

# Evaluate the model

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print("Mean Squared Error:", mse)

print("R-squared:", r2)

* Uses fairly default parameters based on documentation
* Trains science kit Ridge regression model

**Step 13.**

*Selection of the best model*

* Ridge regression was the only one that worked during my scope of the project
* However, it’s a good general use model that still allows us to analyse the data.

**Step 14.**

*Deployment of the Model*

# Scatter plot of actual vs predicted charges

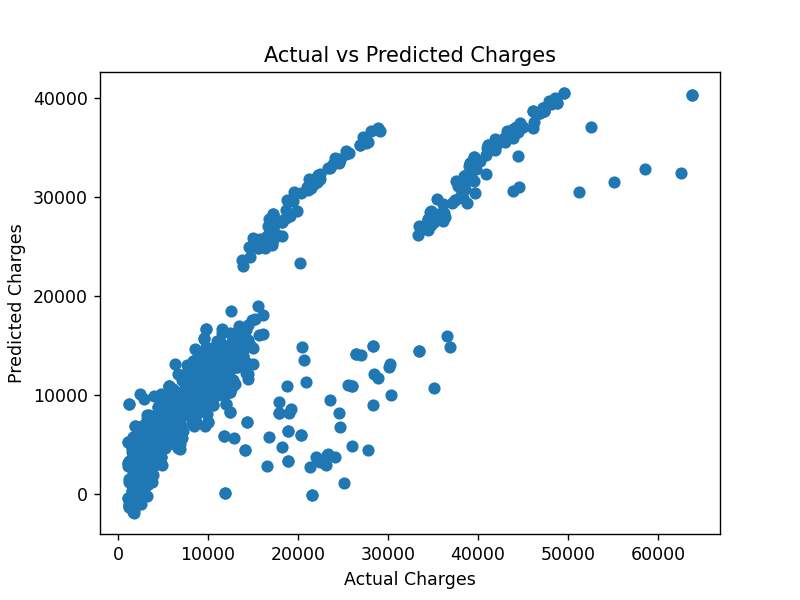
plt.scatter(y\_test, y\_pred)

plt.xlabel("Actual Charges")

plt.ylabel("Predicted Charges")

plt.title("Actual vs Predicted Charges")

plt.show()

* Together with the previous code, this presents us with a scatter plot of the actual vs predicted charges.

**Observations from Step 14.**

* Given our chosen categorical variables of Sex, Smoker and Region, we can see that there is a moderate relationship between the variables.
* As expected of a defined linear relationship, the path is dense
* Actual charges align fairly closely to the predicted charges, although with more variation.

**Step 16.**

*GUI Deployment*

# Root window for the Tkinter GUI

root = tk.Tk()

root.title("Insurance Analysis")

# Generic file loading for flexibility

def load\_data():

    # Open a file dialog to select the CSV file

    file\_path = filedialog.askopenfilename(title="Select CSV File", filetypes=[("CSV Files", "\*.csv")])

    if not file\_path:

        messagebox.showwarning("No File Selected", "Please select a CSV file.")

        return None

    # Loads data

    data = pd.read\_csv(file\_path)

    return data

# Displays data information

def display\_data\_info():

    data = load\_data()

    if data is not None:

        # Display basic info

        info = data.info()

        print(info)  # For console output

        # Show basic stats in a new window

        new\_window = tk.Toplevel(root)

        new\_window.title("Data Information")

        info\_label = tk.Label(new\_window, text=str(data.describe()), justify=tk.LEFT)

        info\_label.pack()

# Create scatter plots

def create\_scatter\_plot():

    data = load\_data()

    if data is not None:

        plt.scatter(data['age'], data['charges'])

        plt.xlabel("Age")

        plt.ylabel("Charges")

        plt.title("Age vs Charges")

        plt.show()

# Define a function to create box plots for categorical data

def create\_box\_plots():

    data = load\_data()

    if data is not None:

        for column in ['sex', 'smoker', 'region']:

            sns.boxplot(x=column, y='charges', data=data)

            plt.title(f"Charges vs {column}")

            plt.show()

# Create buttons

info\_button = tk.Button(root, text="Display Data Information", command=display\_data\_info)

scatter\_button = tk.Button(root, text="Create Scatter Plot", command=create\_scatter\_plot)

box\_plot\_button = tk.Button(root, text="Create Box Plots", command=create\_box\_plots)

# Place the buttons in the GUI

info\_button.pack(pady=10, padx=10)

scatter\_button.pack(pady=10, padx=10)

box\_plot\_button.pack(pady=10, padx=10)

# Start the Tkinter event loop

root.mainloop()

* Creates a Tkinter GUI interface
* Made generic to use on any CSV file
* A screenshot of a computer

  Description automatically generatedIs able to display basic data from the file
* Can present the scatter plot from earlier
* Can go through the sequence of box plots from earlier