Predicting Future Healthcare Reimbursements

Problem Statement:

The healthcare organization aims to develop a predictive model for forecasting Medicare reimbursements. The objective is to enable **proactive resource allocation and budget planning in healthcare organizations**, ultimately leading to informed decision-making and improved patient outcomes. By accurately predicting future reimbursements based on historical data and external factors, the organization seeks to support healthcare providers and policymakers in making informed decisions and enhancing healthcare analytics.

Objective:

The primary objective of this project is to develop a robust predictive model for **forecasting future Medicare reimbursements.** This model will leverage historical data and external factors to accurately predict future reimbursements, thereby enabling strategic decision-making in healthcare organizations. The ultimate goal is to **empower healthcare providers and policymakers to make informed decisions that drive positive change and enhance patient outcomes**. By facilitating** proactive resource allocation and budget planning**, the predictive model aims to contribute to improved healthcare analytics and decision-making processes.

The organization's focus on developing a predictive model aligns with the broader industry trends and the growing importance of data-driven approaches in healthcare. Leveraging advanced analytics and predictive modeling techniques, the organization aims to address the challenges of budget planning and resource allocation, ultimately contributing to improved patient care and operational efficiency.

Installing streamlit library before importing as it not a default library from colab

```
pip install streamlit
```

```
Collecting streamlit
  Downloading streamlit-1.36.0-py2.py3-none-any.whl (8.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.6/8.6 MB 27.0 MB/s eta 0:00:00
Requirement already satisfied: altair<6,>=4.0 in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: blinker<2,>=1.0.0 in /usr/lib/python3/dist-packages (f
Requirement already satisfied: cachetools<6,>=4.0 in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: click<9,>=7.0 in /usr/local/lib/python3.10/dist-packag
```

```
Requirement already satisfied: numpy<3,>=1.20 in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: packaging<25,>=20 in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: pandas<3,>=1.3.0 in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: pillow<11,>=7.1.0 in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: protobuf<6,>=3.20 in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: pyarrow>=7.0 in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: requests<3,>=2.27 in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: rich<14,>=10.14.0 in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: tenacity<9,>=8.1.0 in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: toml<2,>=0.10.1 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: typing-extensions<5,>=4.3.0 in /usr/local/lib/python3.
Collecting gitpython!=3.1.19,<4,>=3.0.7 (from streamlit)
  Downloading GitPython-3.1.43-py3-none-any.whl (207 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.3/207.3 kB 14.8 MB/s eta 0:00:00
Collecting pydeck<1,>=0.8.0b4 (from streamlit)
  Downloading pydeck-0.9.1-py2.py3-none-any.whl (6.9 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6.9/6.9 MB 59.2 MB/s eta 0:00:00
Requirement already satisfied: tornado<7,>=6.0.3 in /usr/local/lib/python3.10/dist-pa
Collecting watchdog<5,>=2.1.5 (from streamlit)
  Downloading watchdog-4.0.1-py3-none-manylinux2014_x86_64.whl (83 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 83.0/83.0 kB 7.8 MB/s eta 0:00:00
Requirement already satisfied: entrypoints in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (fro
Requirement already satisfied: jsonschema>=3.0 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: toolz in /usr/local/lib/python3.10/dist-packages (from
Collecting gitdb<5,>=4.0.1 (from gitpython!=3.1.19,<4,>=3.0.7->streamlit)
  Downloading gitdb-4.0.11-py3-none-any.whl (62 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.7/62.7 kB 7.3 MB/s eta 0:00:00
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/di
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.10/dis
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.10/d
Collecting smmap<6,>=3.0.1 (from gitdb<5,>=4.0.1->gitpython!=3.1.19,<4,>=3.0.7->strea
  Downloading smmap-5.0.1-py3-none-any.whl (24 kB)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/lib
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.10/dist-
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (f
Installing collected packages: watchdog, smmap, pydeck, gitdb, gitpython, streamlit
Successfully installed gitdb-4.0.11 gitpython-3.1.43 pydeck-0.9.1 smmap-5.0.1 streaml
```

```
import streamlit as st
```

1. Import necessary libraries:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVC
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_sco
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import matplotlib.pyplot as plt
import joblib
```

2. Data Pre-processing

Reading a CSV file into a DataFrame using pandas

```
df=pd.read_csv('/content/drive/MyDrive/Cleaned_Data_county (1).csv')
df
```

Finding the shape and size of the dataset

```
df.shape
```

```
(3144, 13)
```

```
df.size
```

```
40872
```

We found 3144 rows and 13 columns in the given dataset. Based on given columns, **12th and 13th columns** which can be **dropped** from the entire table as it **contains null values**. Reducing the size of the dataset will increase the speed of training and testing process especially applied when dataset is big. Will **contribute more in increasing the accuracy of the model**.

No of cells is **40872** before dropping the null values

3.Dimensionality reduction

Dropping the columns with only null values

```
df=df.drop(['Unnamed: 10','Unnamed: 11','Unnamed: 12'],axis=1)
```

```
df=df.drop(['Unnamed: 10','Unnamed: 11','Unnamed: 12'],axis=1)
df.head()
```

Shape and size of data after removing columns with null values

```
df.shape
```

```
(3144, 10)
```

```
df.size
```

```
31440
```

Now the number of rows and columns got **reduced** to **3144 rows and 10 columns**

No of cells **reduced** to **31440**

The reduction of **data size by 23%** after removing columns with null values is a critical step in our data preprocessing, as it not only optimizes the **computational efficiency** of our analysis but also ensures that our insights are based on a more complete and **reliable dataset**. This streamlined approach allows for more focused and **accurate modeling and analysis**, ultimately

leading to more robust and meaningful results.

Exploring Data Types Using Key-Value Grouping

```
for k,v in df.groupby(df.dtypes,axis=1):
  print(k, v.columns)
```

```
int64 Index(['County ID'], dtype='object')
float64 Index(['Durable medical equipment reimbursements per enrollee (2014)'], dtype
object Index(['County name', 'Medicare enrollees (2014)',
       'Total Medicare reimbursements per enrollee (2014)',
       'Hospital & skilled nursing facility reimbursements per enrollee (2014)',
       'Physician reimbursements per enrollee (2014)',
       'Outpatient facility reimbursements per enrollee (2014)',
       'Home health agency reimbursements per enrollee (2014)',
       'Hospice reimbursements per enrollee (2014)'],
      dtype='object')
```

```
columns = ['Medicare enrollees (2014)',
    'Total Medicare reimbursements per enrollee (2014)',
       'Hospital & skilled nursing facility reimbursements per enrollee (2014)',
       'Physician reimbursements per enrollee (2014)',
       'Outpatient facility reimbursements per enrollee (2014)',
       'Home health agency reimbursements per enrollee (2014)',
      'Hospice reimbursements per enrollee (2014)']

data_types = df[columns].dtypes
data_types
```

```
Medicare enrollees (2014)                                          object
Total Medicare reimbursements per enrollee (2014)                  object
Hospital & skilled nursing facility reimbursements per enrollee (2014)    object
Physician reimbursements per enrollee (2014)                       object
Outpatient facility reimbursements per enrollee (2014)             object
Home health agency reimbursements per enrollee (2014)              object
Hospice reimbursements per enrollee (2014)                         object
dtype: object
```

The columns pertaining to Medicare enrollees and reimbursements, despite **containing numerical values**, are currently **represented as string data types** when examining the data types. This discrepancy may arise from the presence of non-numeric characters or formatting issues within the data. To address this, it is crucial to ensure that the **data is accurately represented as numerical values for effective analysis and interpretation**. By employing appropriate data manipulation techniques, such as **removing non-numeric characters** and converting the data type to numeric, the integrity of the data can be preserved, allowing for precise and meaningful analysis. This process involves meticulous attention to detail and

precise and meaningful analysis. This process involves meticulous attention to detail and adherence to best practices in data handling to **ensure the accurate representation of the numerical values within the specified columns.**

The reason for showing it as string is becuase of the existence of "," in each column.

Let's **remove "," from each column to convert them to numerical values**.

```
for column in columns:
    df[column] = pd.to_numeric(df[column].str.replace(',', ''), errors='coerce')

# Check the data types again
data_types = df[columns].dtypes
data_types
```

```
Medicare enrollees (2014)                                          int64
Total Medicare reimbursements per enrollee (2014)                float64
Hospital & skilled nursing facility reimbursements per enrollee (2014)    float64
Physician reimbursements per enrollee (2014)                     float64
Outpatient facility reimbursements per enrollee (2014)           float64
Home health agency reimbursements per enrollee (2014)            float64
Hospice reimbursements per enrollee (2014)                       float64
dtype: object
```

After **removing the comma**, the output is returned as a float data type. This means that the result of the **operation is being interpreted as a floating-point number in the programming context**. When a comma is removed from a numerical value, it typically signifies a **change from a string or other data type to a numerical value**, and in this case, a float data type is returned.

```
df.describe()
```

```
for k,v in df.groupby(df.dtypes,axis=1):
  print(k, v.columns)
```

```
    int64 Index(['County ID', 'Medicare enrollees (2014)'], dtype='object')
    float64 Index(['Total Medicare reimbursements per enrollee (2014)',
           'Hospital & skilled nursing facility reimbursements per enrollee (2014)',
           'Physician reimbursements per enrollee (2014)',
           'Outpatient facility reimbursements per enrollee (2014)',
           'Home health agency reimbursements per enrollee (2014)',
           'Hospice reimbursements per enrollee (2014)',
           'Durable medical equipment reimbursements per enrollee (2014)'],
          dtype='object')
    object Index(['County name'], dtype='object')
```

When handling float values in the specified columns, it is essential to ensure **accurate representation without decimal points**. To achieve this, the null values within the columns can be **replaced with the median value based on given nature of this dataset**, and subsequently, the **float values can be converted to integers**. This process involves utilizing the **fillna()** method to replace null values with the median, followed by the use of the** astype(int)** method to convert the float values to integers. By implementing this approach, the **integrity of the data is maintained**, and the columns are effectively handled to reflect integer values, thus ensuring precision and accuracy in the representation of the data.

```
columns_to_convert = ['Total Medicare reimbursements per enrollee (2014)',
                      'Hospital & skilled nursing facility reimbursements per enrollee (2
                      'Physician reimbursements per enrollee (2014)',
                      'Outpatient facility reimbursements per enrollee (2014)',
                      'Home health agency reimbursements per enrollee (2014)',
                      'Hospice reimbursements per enrollee (2014)',
                      'Durable medical equipment reimbursements per enrollee (2014)']

# Treat null values with the median
df[columns_to_convert] = df[columns_to_convert].fillna(df[columns_to_convert].median())

# Convert float values to integers
df[columns_to_convert] = df[columns_to_convert].astype(int)
```

```
for k,v in df.groupby(df.dtypes,axis=1):
  print(k, v.columns)
```

```
    int64 Index(['County ID', 'Medicare enrollees (2014)',
           'Total Medicare reimbursements per enrollee (2014)',
           'Hospital & skilled nursing facility reimbursements per enrollee (2014)',
           'Physician reimbursements per enrollee (2014)',
```

```
            'Outpatient facility reimbursements per enrollee (2014)',
            'Home health agency reimbursements per enrollee (2014)',
            'Hospice reimbursements per enrollee (2014)',
            'Durable medical equipment reimbursements per enrollee (2014)'],
           dtype='object')
     object Index(['County name'], dtype='object')
```

```
pd.set_option('display.float_format', lambda x: '%.0f' % x)
```

Checking null values in each column

```
df.isnull().sum()
```

```
County ID                                                          0
County name                                                        0
Medicare enrollees (2014)                                          0
Total Medicare reimbursements per enrollee (2014)                  0
Hospital & skilled nursing facility reimbursements per enrollee (2014)    0
Physician reimbursements per enrollee (2014)                       0
Outpatient facility reimbursements per enrollee (2014)             0
Home health agency reimbursements per enrollee (2014)              0
Hospice reimbursements per enrollee (2014)                         0
Durable medical equipment reimbursements per enrollee (2014)       0
dtype: int64
```

```
df.describe()
```

Dealing with Outliers

```
columns_to_check = ['Medicare enrollees (2014)', 'Total Medicare reimbursements per enrol
                    'Hospital & skilled nursing facility reimbursements per enrollee (201
                    'Physician reimbursements per enrollee (2014)', 'Outpatient facility
                    'Home health agency reimbursements per enrollee (2014)', 'Hospice rei
                    'Durable medical equipment reimbursements per enrollee (2014)']
plt.figure(figsize=(10, 6))
df[columns_to_check].boxplot()
plt.title('Boxplot of Medicare Reimbursements')
plt.xticks(rotation=45)
plt.show()
```

Here, we have one outlier at mediacre enrollees column. So let's drop the last row.

```
df = df.iloc[:-1]
```

Let's check if the outlier exists.

```
df.describe()
```

```
columns_to_check = ['Medicare enrollees (2014)', 'Total Medicare reimbursements per enrol
                    'Hospital & skilled nursing facility reimbursements per enrollee (201
                    'Physician reimbursements per enrollee (2014)', 'Outpatient facility
                    'Home health agency reimbursements per enrollee (2014)', 'Hospice rei
                    'Durable medical equipment reimbursements per enrollee (2014)']
plt.figure(figsize=(10, 6))
df[columns_to_check].boxplot()
plt.title('Boxplot of Medicare Reimbursements')
plt.xticks(rotation=45)
plt.show()
```

```
# Calculate Q1 (25th percentile) and Q3 (75th percentile)
Q1 = df['Medicare enrollees (2014)'].quantile(0.25)
Q3 = df['Medicare enrollees (2014)'].quantile(0.75)
```

```
# Calculate IQR
IQR = Q3 - Q1

# Define the lower and upper bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Identify outliers
outliers = df[(df['Medicare enrollees (2014)'] < lower_bound) | (df['Medicare enrollees (
outliers
```

Replacing outliers with median values.

In medicare enrolees column, we have 357 outliers which could be treated with the help of median.

```
median_enrollees = df['Medicare enrollees (2014)'].median()

# Replace outliers with the median value
df['Medicare enrollees (2014)'] = df['Medicare enrollees (2014)'].apply(lambda x: median_


df.describe()
```

Downloading the pre-processed dataset

```
from google.colab import files

df.to_csv('cleaned_data.csv')
files.download('cleaned_data.csv')
```

3.Exploratory data analysis

After pre-processing the dataset, let's use the **treated dataset** without null values and outliers treatment.

```
pd.set_option('display.float_format', lambda x: '%.0f' % x)
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVC
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_sco
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import matplotlib.pyplot as plt
import joblib
```

```
df=pd.read_csv('/content/Beni_cleaned_data.csv')
df
```

```
df.describe()
```

Pie Chart Analysis of Medicare Reimbursements by Healthcare Service Category

```
# Define data
labels = ['Hospital & skilled nursing facility reimbursements per enrollee (2014)',
          'Physician reimbursements per enrollee (2014)',
          'Outpatient facility reimbursements per enrollee (2014)',
          'Home health agency reimbursements per enrollee (2014)',
          'Hospice reimbursements per enrollee (2014)',
          'Durable medical equipment reimbursements per enrollee (2014)']
```

```
# Get values from the DataFrame
values = df[['Hospital & skilled nursing facility reimbursements per enrollee (2014)',
           'Physician reimbursements per enrollee (2014)',
           'Outpatient facility reimbursements per enrollee (2014)',
           'Home health agency reimbursements per enrollee (2014)',
           'Hospice reimbursements per enrollee (2014)',
           'Durable medical equipment reimbursements per enrollee (2014)']].sum().values

# Create a pie chart
plt.pie(values, labels=labels, autopct='%1.1f%%')
plt.title('Distribution of Medicare Reimbursements by Category')
plt.show()
```

Insights found from plotting a pie chart to anlayse each categories of reimburesemens per enrollee and their contribution towards Total Medicare reimbursements per enrollee:

1.Hospital & Skilled Nursing Facility Reimbursements: A 1% increase in Hospital & skilled nursing facility reimbursements per enrollee (2014) is associated with a **45.1% increase** in Total Medicare reimbursements.

2.Physician Reimbursements: A 1% increase in Physician reimbursements per enrollee (2014) corresponds to a **23.4% increase** in Total Medicare reimbursements.

3.Outpatient Facility Reimbursements: There is a notable relationship, where a 1% increase in Outpatient facility reimbursements per enrollee (2014) is linked to a **20.2% increase** in Total Medicare reimbursements.

4.Home Health Agency Reimbursements: A 1% increase in Home health agency reimbursements per enrollee (2014) corresponds to a **5.2% increase** in Total Medicare reimbursements.

5.Hospice Reimbursements: There is a discernible relationship, where a 1% increase in Hospice reimbursements per enrollee (2014) is associated with a **4% increase** in Total Medicare reimbursements.

6.Durable Medical Equipment Reimbursements: A 1% increase in Durable medical equipment reimbursements per enrollee (2014) corresponds to a **2.2% increase** in Total Medicare reimbursements.

Correlation Matrix Analysis and its Role in Assessing Relationships Between Predictor and Target Variables

In order to comprehensively understand the relationship between each predictor variable and the target variable, a correlation matrix was constructed to assess the strength and direction of these associations. The correlation matrix revealed valuable **insights into the degree of linear dependence between the predictor variables,** encompassing "Hospital & skilled nursing facility reimbursements per enrollee (2014)", "Physician reimbursements per enrollee (2014)", "Outpatient facility reimbursements per enrollee (2014)", "Home health agency reimbursements per enrollee (2014)", "Hospice reimbursements per enrollee (2014)", and "Durable medical equipment reimbursements per enrollee (2014)", **with the target variable** "Total Medicare reimbursements per enrollee (2014)". Each correlation coefficient **signifies the extent to which a change in one variable coincides with a change in the target variable**, thereby providing essential quantitative insights for informed decision-making. This analysis elucidates the nuanced interplay between the predictor variables and the target variable, offering a robust foundation for subsequent data-driven interpretations and strategic endeavors.

```
df.set_index('County name', inplace=True)
```

```
plt.figure(figsize=(10, 6))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```

```
plt.savefig('correlation.png')
```

Insights to guide stakeholders in making informed investment decisions:

1.Hospital & Skilled Nursing Facility Reimbursements:

- A **0.9% increase** in Hospital & Skilled Nursing Facility reimbursements per enrollee (2014) is associated with a **45.1% increase** in Total Medicare reimbursements.
- This indicates a **strong positive** correlation between Hospital & Skilled Nursing Facility reimbursements and Total Medicare reimbursements, suggesting that **investing in this area** can yield significant returns in terms of Medicare reimbursements.

2.Physician Reimbursements:

- A **0.49% increase** in Physician reimbursements per enrollee (2014) corresponds to a **23.4% increase** in Total Medicare reimbursements.
- This demonstrates a **substantial positive** correlation between Physician reimbursements and Total Medicare reimbursements, highlighting the potential for favorable returns on investment in physician services.

3.Outpatient Facility Reimbursements:

- A **0.17% increase** in Outpatient Facility reimbursements per enrollee (2014) is linked to a **20.2% increase** in Total Medicare reimbursements.
- This signifies a notable relationship between Outpatient Facility reimbursements and Total Medicare reimbursements, indicating the potential for **increased profitability through strategic investments** in outpatient facilities.

4.Home Health Agency Reimbursements:

- A 0.62% increase in Home Health Agency reimbursements per enrollee (2014) corresponds to a 5.2% increase in Total Medicare reimbursements. While the correlation between Home Health Agency reimbursements and
- Total Medicare reimbursements is positive, the impact is relatively lower compared to other variables, suggesting a moderate potential for increased profitability in this area.

5.Hospice Reimbursements:

- A 0.3% increase in Hospice reimbursements per enrollee (2014) is associated with a 4% increase in Total Medicare reimbursements.
- The correlation between Hospice reimbursements and Total Medicare reimbursements

indicates a discernible relationship, albeit with a relatively lower impact on overall Medicare reimbursements.

6.Durable Medical Equipment Reimbursements:

- A 0.36% increase in Durable Medical Equipment reimbursements per enrollee (2014) corresponds to a 2.2% increase in Total Medicare reimbursements.
- The correlation between Durable Medical Equipment reimbursements and Total Medicare reimbursements suggests a relatively lower impact on overall Medicare reimbursements compared to other variables.

These insights provide valuable guidance for stakeholders, indicating the potential for **increased profitability** by **strategically investing in Hospital & Skilled Nursing Facility reimbursements, Physician reimbursements, and Outpatient Facility reimbursements**. While Home Health Agency, Hospice, and Durable Medical Equipment reimbursements also demonstrate positive correlations with Total Medicare reimbursements, their impact may be comparatively lower.

Selecting features and target variable

```
X = df[['Medicare enrollees (2014)', 'Hospital & skilled nursing facility reimbursements
        'Physician reimbursements per enrollee (2014)', 'Outpatient facility reimbursemen
        'Home health agency reimbursements per enrollee (2014)', 'Hospice reimbursements
        'Durable medical equipment reimbursements per enrollee (2014)']]
y = df['Total Medicare reimbursements per enrollee (2014)']


# Splitting the dataset 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)
```

4.Choosing a right Machine learning model with best accuracy

Multiple linear regression

```
# Initializing the linear regression model
model = LinearRegression()


# Training the model
model.fit(X_train, y_train)
```

```
model.fit(X_train, y_train)
```

```
# Making predictions
y_pred = model.predict(X_test)
```

```
# Visualizing the actual vs. predicted values
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 6))
sns.scatterplot(x=y_test, y=y_pred, color='red', label='Predicted')
sns.scatterplot(x=y_test, y=y_test, color='blue', label='Actual')
plt.xlabel('Actual Total Medicare reimbursements per enrollee (2014)')
plt.ylabel('Predicted Total Medicare reimbursements per enrollee (2014)')
plt.title('Actual vs. Predicted Values')
plt.legend()
plt.show()
```

```
accuracy = model.score(X_test, y_test)

# Print the accuracy
print("Accuracy:", accuracy)
```

```
Accuracy: 0.9774135252268249
```

```
# Calculate R-squared and MSE
r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)

# Print the results
print("R-squared:", r2)
print("MSE:", mse)
```

```
R-squared: 0.9774135252268249
MSE: 50850.077243634965
```

Random Forest regression model

```
# Initializing the Random Forest regression model
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)


# Training the model
rf_model.fit(X_train, y_train)
```

```
# Making predictions
y_pred_rf = rf_model.predict(X_test)


# Create a figure and axes
fig, ax = plt.subplots(figsize=(10, 6))

# Plot the actual values
sns.scatterplot(x=y_test, y=y_pred_rf, ax=ax, color='red', label='Actual')

# Plot the predicted values
sns.scatterplot(x=y_test, y=y_pred, ax=ax, color='blue', label='Predicted')

# Set the labels and title
```

```
# Set the labels and title
ax.set_xlabel('Actual Total Medicare reimbursements per enrollee (2014)')
ax.set_ylabel('Predicted Total Medicare reimbursements per enrollee (2014)')
ax.set_title('Actual vs. Predicted Values (Random Forest)')

# Add a legend
ax.legend()

# Show the plot
plt.show()
```

```
# Calculate the accuracy of the Random Forest model
accuracy_rf = rf_model.score(X_test, y_test)


# Print the accuracy
print("Random Forest Model Accuracy:", accuracy_rf)
```

```
Random Forest Model Accuracy: 0.9533613809192207
```

Ridge Regression model

```
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


preprocessor = Pipeline(steps=[('scaler', StandardScaler())])


# Ridge Regression pipeline
ridge_pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('regressor', Ridge(alpha=1.0))  # Adjust alpha as needed
])


# Train Ridge Regression model
ridge_pipeline.fit(X_train, y_train)
ridge_pred = ridge_pipeline.predict(X_test)


# Evaluate models
def evaluate_model(name, y_test, y_pred):
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    print(f'{name} Regression MSE: {mse}')
    print(f'{name} Regression R2 Score: {r2}\n')


# Create a figure and axes
fig, ax = plt.subplots(figsize=(10, 6))

# Plot the actual values
sns.scatterplot(x=y_test, y=y_pred_rf, ax=ax, color='red', label='Actual')

# Plot the predicted values
sns.scatterplot(x=y_test, y=ridge_pred, ax=ax, color='blue', label='Predicted')

# Set the labels and title
ax.set_xlabel('Actual Total Medicare reimbursements per enrollee (2014)')
ax.set_ylabel('Predicted Total Medicare reimbursements per enrollee (2014)')
ax.set_title('Actual vs. Predicted Values (Ridge Regression)')

# Add a legend
ax.legend()

# Show the plot
plt.show()
```

```
evaluate_model('Ridge', y_test, ridge_pred)
```

```
Ridge Regression MSE: 50867.15915916109
Ridge Regression R2 Score: 0.9774059378193907
```

```
print(f'Ridge Regression Accuracy: {ridge_pipeline.score(X_test, y_test)}')
```

```
Ridge Regression Accuracy: 0.9774059378193907
```

Lasso Regression model

```
# Lasso Regression pipeline
lasso_pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
```

```
    ('regressor', Lasso(alpha=0.1))  # Adjust alpha as needed
])


# Train Lasso Regression model
lasso_pipeline.fit(X_train, y_train)
lasso_pred = lasso_pipeline.predict(X_test)


# Evaluate models
def evaluate_model(name, y_test, y_pred):
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    print(f'{name} Regression MSE: {mse}')
    print(f'{name} Regression R2 Score: {r2}\n')


# Create a figure and axes
fig, ax = plt.subplots(figsize=(10, 6))

# Plot the actual values
sns.scatterplot(x=y_test, y=y_pred_rf, ax=ax, color='red', label='Actual')

# Plot the predicted values
sns.scatterplot(x=y_test, y=ridge_pred, ax=ax, color='blue', label='Predicted')

# Set the labels and title
ax.set_xlabel('Actual Total Medicare reimbursements per enrollee (2014)')
ax.set_ylabel('Predicted Total Medicare reimbursements per enrollee (2014)')
ax.set_title('Actual vs. Predicted Values (Ridge Regression)')

# Add a legend
ax.legend()

# Show the plot
plt.show()
```

```
evaluate_model('Lasso', y_test, lasso_pred)
```

```
     Lasso Regression MSE: 50865.10761290849
     Lasso Regression R2 Score: 0.9774068490706649
```

```
print(f'Lasso Regression Accuracy: {lasso_pipeline.score(X_test, y_test)}')
```

```
     Lasso Regression Accuracy: 0.9774068490706649
```

Comparing accuracy of ML models

```
import plotly.graph_objects as go

# Sample data
x = ['Multiple linear regression', 'Random Forest regression model', 'Ridge Regression mo
y = [0.9774135, 0.9533613, 0.9774059, 0.9774068]

# Define colors for the bars
colors = ['red', 'blue', 'green', 'hotpink']

# Creating the bar chart with colors and animations
fig = go.Figure(data=[go.Bar(x=x, y=y, marker_color=colors, marker_line_color='black', ma

# Add animation
fig.update_layout(updatemenus=[dict(type='buttons', showactive=False,
                                    buttons=[dict(label='Play',
                                                  method='animate',
                                                  args=[None, dict(frame=dict(duration=50
                                                                   fromcurrent=True)])])])
```

```
# Define the frames for the animation
frames = [dict(data=[dict(type='bar',
                          x=x,
                          y=[0.5, 0.3, 0.6, 0.4],
                          marker_color=colors,
                          marker_line_color='black',
                          marker_line_width=2)]),
          dict(data=[dict(type='bar',
                          x=x,
                          y=[0.5, 0.6, 0.1, 0.6],
                          marker_color=colors,
                          marker_line_color='black',
                          marker_line_width=2)]),
          dict(data=[dict(type='bar',
                          x=x,
                          y=[0.9774135, 0.9533613, 0.9774059, 0.9774068],
                          marker_color=colors,
                          marker_line_color='black',
                          marker_line_width=2)])]

# Add frames to the animation
fig.frames = frames

# Display the chart
fig.show()
```

5.Model deployment

```
pip install streamlit
```

```
Collecting streamlit
  Downloading streamlit-1.46.1-py3-none-any.whl.metadata (9.0 kB)
Requirement already satisfied: altair<6,>=4.0 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: blinker<2,>=1.5.0 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: cachetools<7,>=4.0 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: click<9,>=7.0 in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: numpy<3,>=1.23 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: packaging<26,>=20 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: pandas<3,>=1.4.0 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: pillow<12,>=7.1.0 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: protobuf<7,>=3.20 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: pyarrow>=7.0 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: requests<3,>=2.27 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: tenacity<10,>=8.1.0 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: toml<2,>=0.10.1 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: typing-extensions<5,>=4.4.0 in /usr/local/lib/python3.
Collecting watchdog<7,>=2.1.5 (from streamlit)
  Downloading watchdog-6.0.0-py3-none-manylinux2014_x86_64.whl.metadata (44 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 44.3/44.3 kB 1.9 MB/s eta 0:00:00
Requirement already satisfied: gitpython!=3.1.19,<4,>=3.0.7 in /usr/local/lib/python3
Collecting pydeck<1,>=0.8.0b4 (from streamlit)
  Downloading pydeck-0.9.1-py2.py3-none-any.whl.metadata (4.1 kB)
Requirement already satisfied: tornado!=6.5.0,<7,>=6.0.3 in /usr/local/lib/python3.11
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (fro
Requirement already satisfied: jsonschema>=3.0 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: narwhals>=1.14.2 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: gitdb<5,>=4.0.1 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/di
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: smmap<6,>=3.0.1 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/lib
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (f
Downloading streamlit-1.46.1-py3-none-any.whl (10.1 MB)
```

```
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 10.1/10.1 MB 58.1 MB/s eta 0:00:00
Downloading pydeck-0.9.1-py2.py3-none-any.whl (6.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6.9/6.9 MB 84.3 MB/s eta 0:00:00
Downloading watchdog-6.0.0-py3-none-manylinux2014_x86_64.whl (79 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 79.1/79.1 kB 6.5 MB/s eta 0:00:00
Installing collected packages: watchdog, pydeck, streamlit
Successfully installed pydeck-0.9.1 streamlit-1.46.1 watchdog-6.0.0
```

```
import streamlit as st
import joblib


# Write the Streamlit app code to a file
#Manually enter values
%%writefile app.py

import streamlit as st
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error
from sklearn.model_selection import train_test_split


st.markdown("""
    <style>
    .main {
        background: url('/content/health.jpg') no-repeat center center fixed;
        background-size: cover;
        color: red;
    }
    .title {
        font-family: 'Arial', sans-serif;
        color: #FF5733;
        text-align: center;
        margin-top: 30px;
    }
    .text-input {
        background: rgba(255, 255, 255, 0.7);
        border-radius: 5px;
        padding: 10px;
        margin: 10px 0;
    }
    </style>
    """, unsafe_allow_html=True)

# Read the dataset from the uploaded CSV file

df = pd.read_csv('/content/Beni_cleaned_data.csv')

# Display the first few rows of the dataframe for verification
```

```
st.write("Sample Data:")
st.write(df.head())

# Update columns according to your dataset
# Assuming your dataset has the following columns:
# 'Medicare enrollees (2014)', 'Hospital & skilled nursing facility reimbursements per en
# 'Physician reimbursements per enrollee (2014)', 'Outpatient facility reimbursements per
# 'Home health agency reimbursements per enrollee (2014)', 'Hospice reimbursements per en
# 'Durable medical equipment reimbursements per enrollee (2014)', 'Total Medicare reimbur

# Splitting the dataset
X = df[['Medicare enrollees (2014)', 'Hospital & skilled nursing facility reimbursements
        'Physician reimbursements per enrollee (2014)', 'Outpatient facility reimbursemen
        'Home health agency reimbursements per enrollee (2014)', 'Hospice reimbursements
        'Durable medical equipment reimbursements per enrollee (2014)']]
y = df['Total Medicare reimbursements per enrollee (2014)']

# Splitting 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)

# Initializing and training the model
model = LinearRegression()
model.fit(X_train, y_train)

# Define a function for making predictions
def predict_medicare_reimbursements(input_data):
    prediction = model.predict(np.array(input_data).reshape(1, -1))
    return prediction[0]

# Streamlit app
st.title("Medicare Reimbursement Predictor")
st.write("Enter the details to predict the total Medicare reimbursements per enrollee.")

# User input for each feature
enrollees = st.number_input('Medicare enrollees (2014)', min_value=0)
hospital_reimbursements = st.number_input('Hospital & skilled nursing facility reimbursem
physician_reimbursements = st.number_input('Physician reimbursements per enrollee (2014)'
outpatient_reimbursements = st.number_input('Outpatient facility reimbursements per enrol
home_health_reimbursements = st.number_input('Home health agency reimbursements per enrol
hospice_reimbursements = st.number_input('Hospice reimbursements per enrollee (2014)', mi
durable_equipment_reimbursements = st.number_input('Durable medical equipment reimburseme

# Make a prediction when the user clicks the button
if st.button("Predict"):
    input_data = [enrollees, hospital_reimbursements, physician_reimbursements, outpatien
                  home_health_reimbursements, hospice_reimbursements, durable_equipment_r

    # Validate input
    if not any(input_data):
        st.error("Please enter valid inputs for all fields.")
```

```
    else:
        prediction = predict_medicare_reimbursements(input_data)
        st.success(f"The predicted total Medicare reimbursements per enrollee is: ${predi

# Display model performance on the test set
st.write("Model Performance on Test Data:")
y_pred = model.predict(X_test)
r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
st.write(f"R-squared: {r2:.2f}")
st.write(f"Mean Squared Error: {mse:.2f}")
```

```
 Overwriting app.py
```

```
# Write the Streamlit app code to a file
%%writefile app.py

import streamlit as st
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error
from sklearn.model_selection import train_test_split


st.markdown("""
    <style>
    .main {
        backgroundcolor:#4CAF50;
        background-size: cover;
        color: red;
    }
    .title {
        font-family: 'Arial', sans-serif;
        color: #FF5733;
        text-align: center;
        margin-top: 30px;
    }
    .text-input {
        background: rgba(255, 255, 255, 0.7);
        border-radius: 5px;
        padding: 10px;
        margin: 10px 0;
    }
    </style>
    """, unsafe_allow_html=True)

# Read the dataset from the uploaded CSV file

df = pd.read_csv('/content/Beni_cleaned_data.csv')
```

```
# Display the first few rows of the dataframe for verification
st.write("Sample Data:")
st.write(df.head())

# Update columns according to your dataset
# Assuming your dataset has the following columns:
# 'Medicare enrollees (2014)', 'Hospital & skilled nursing facility reimbursements per en
# 'Physician reimbursements per enrollee (2014)', 'Outpatient facility reimbursements per
# 'Home health agency reimbursements per enrollee (2014)', 'Hospice reimbursements per en
# 'Durable medical equipment reimbursements per enrollee (2014)', 'Total Medicare reimbur

# Splitting the dataset
X = df[['Medicare enrollees (2014)', 'Hospital & skilled nursing facility reimbursements
        'Physician reimbursements per enrollee (2014)', 'Outpatient facility reimbursemen
        'Home health agency reimbursements per enrollee (2014)', 'Hospice reimbursements
        'Durable medical equipment reimbursements per enrollee (2014)']]
y = df['Total Medicare reimbursements per enrollee (2014)']

# Splitting 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)

# Initializing and training the model
model = LinearRegression()
model.fit(X_train, y_train)

# Define a function for making predictions
def predict_medicare_reimbursements(input_data):
    prediction = model.predict(np.array(input_data).reshape(1, -1))
    return prediction[0]

# Streamlit app
st.title("Medicare Reimbursement Predictor")
st.write("Enter the details to predict the total Medicare reimbursements per enrollee.")

# User input for each feature
enrollees = st.number_input('Medicare enrollees (2014)', min_value=0)
hospital_reimbursements = st.number_input('Hospital & skilled nursing facility reimbursem
physician_reimbursements = st.number_input('Physician reimbursements per enrollee (2014)'
outpatient_reimbursements = st.number_input('Outpatient facility reimbursements per enrol
home_health_reimbursements = st.number_input('Home health agency reimbursements per enrol
hospice_reimbursements = st.number_input('Hospice reimbursements per enrollee (2014)', mi
durable_equipment_reimbursements = st.number_input('Durable medical equipment reimburseme

# Make a prediction when the user clicks the button
if st.button("Predict"):
    input_data = [enrollees, hospital_reimbursements, physician_reimbursements, outpatien
                  home_health_reimbursements, hospice_reimbursements, durable_equipment_r

    # Validate input
    if not any(input_data):
```

```
            st.error("Please enter valid inputs for all fields.")
        else:
            prediction = predict_medicare_reimbursements(input_data)
            st.success(f"The predicted total Medicare reimbursements per enrollee is: ${predi

    # Display model performance on the test set
    st.write("Model Performance on Test Data:")
    y_pred = model.predict(X_test)
    r2 = r2_score(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)
    st.write(f"R-squared: {r2:.2f}")
    st.write(f"Mean Squared Error: {mse:.2f}")


    Overwriting app.py


    # Save the trained model
    joblib.dump(model, 'healthcare_prediction_model.joblib')

    ['healthcare_prediction_model.joblib']


    Start coding or generate with AI.


    #loading the trained model
    loaded_model = joblib.load('healthcare_prediction_model.joblib')


    !npm install localtunnel

    ⠙⠹⠸⠼⠴⠦⠧⠇⠏
    up to date, audited 23 packages in 914ms
    ⠇
    ⠇3 packages are looking for funding
    ⠇  run `npm fund` for details
    ⠇
    2 high severity vulnerabilities

    To address all issues (including breaking changes), run:
      npm audit fix --force

    Run `npm audit` for details.
    ⠇


    !wget -q -O - https://loca.lt/mytunnelpassword

    34.23.140.174


    !streamlit run app.py &>/content/logs.txt & npx localtunnel --port 8501 and curl ipv4.ica
```

```
your url is: https://ready-parents-smash.loca.lt
^C
```

```
!pip install streamlit pyngrok
```

```
Requirement already satisfied: streamlit in /usr/local/lib/python3.10/dist-packages (
Collecting pyngrok
  Downloading pyngrok-7.1.6-py3-none-any.whl (22 kB)
Requirement already satisfied: altair<6,>=4.0 in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: blinker<2,>=1.0.0 in /usr/lib/python3/dist-packages (f
Requirement already satisfied: cachetools<6,>=4.0 in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: click<9,>=7.0 in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: numpy<3,>=1.20 in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: packaging<25,>=20 in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: pandas<3,>=1.3.0 in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: pillow<11,>=7.1.0 in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: protobuf<6,>=3.20 in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: pyarrow>=7.0 in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: requests<3,>=2.27 in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: rich<14,>=10.14.0 in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: tenacity<9,>=8.1.0 in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: toml<2,>=0.10.1 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: typing-extensions<5,>=4.3.0 in /usr/local/lib/python3.
Requirement already satisfied: gitpython!=3.1.19,<4,>=3.0.7 in /usr/local/lib/python3
Requirement already satisfied: pydeck<1,>=0.8.0b4 in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: tornado<7,>=6.0.3 in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: watchdog<5,>=2.1.5 in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: PyYAML>=5.1 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: entrypoints in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (fro
Requirement already satisfied: jsonschema>=3.0 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: toolz in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: gitdb<5,>=4.0.1 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/di
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.10/dis
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.10/d
Requirement already satisfied: smmap<6,>=3.0.1 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/lib
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.10/dist-
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (f
Installing collected packages: pyngrok
Successfully installed pyngrok-7.1.6
```

```
# Write the Streamlit app code to a file
%%writefile app.py

import streamlit as st
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error
from sklearn.model_selection import train_test_split


st.markdown("""
    <style>
    .main {
        background: url('/content/health.jpg') no-repeat center center fixed;
        background-size: cover;
        color: red;
    }
    .title {
        font-family: 'Arial', sans-serif;
        color: #FF5733;
        text-align: center;
        margin-top: 30px;
    }
    .text-input {
        background: rgba(255, 255, 255, 0.7);
        border-radius: 5px;
        padding: 10px;
        margin: 10px 0;
    }
    </style>
    """, unsafe_allow_html=True)

# Read the dataset from the uploaded CSV file

df = pd.read_csv('/content/drive/MyDrive/cleaned_data.csv')

# Display the first few rows of the dataframe for verification
st.write("Sample Data:")
st.write(df.head())

# Update columns according to your dataset
# Assuming your dataset has the following columns:
# 'Medicare enrollees (2014)', 'Hospital & skilled nursing facility reimbursements per en
# 'Physician reimbursements per enrollee (2014)', 'Outpatient facility reimbursements per
# 'Home health agency reimbursements per enrollee (2014)', 'Hospice reimbursements per en
# 'Durable medical equipment reimbursements per enrollee (2014)', 'Total Medicare reimbur

# Splitting the dataset
X = df[['Medicare enrollees (2014)', 'Hospital & skilled nursing facility reimbursements
        'Physician reimbursements per enrollee (2014)', 'Outpatient facility reimbursemen
```

```
        'Home health agency reimbursements per enrollee (2014)', 'Hospice reimbursements
        'Durable medical equipment reimbursements per enrollee (2014)']]
y = df['Total Medicare reimbursements per enrollee (2014)']

# Splitting 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)

# Initializing and training the model
model = LinearRegression()
model.fit(X_train, y_train)

# Define a function for making predictions
def predict_medicare_reimbursements(input_data):
    prediction = model.predict(np.array(input_data).reshape(1, -1))
    return prediction[0]

# Streamlit app
st.title("Medicare Reimbursement Predictor")
st.write("Enter the details to predict the total Medicare reimbursements per enrollee.")

# User input for each feature
enrollees = st.number_input('Medicare enrollees (2014)', min_value=0)
hospital_reimbursements = st.number_input('Hospital & skilled nursing facility reimbursem
physician_reimbursements = st.number_input('Physician reimbursements per enrollee (2014)'
outpatient_reimbursements = st.number_input('Outpatient facility reimbursements per enrol
home_health_reimbursements = st.number_input('Home health agency reimbursements per enrol
hospice_reimbursements = st.number_input('Hospice reimbursements per enrollee (2014)', mi
durable_equipment_reimbursements = st.number_input('Durable medical equipment reimburseme

# Make a prediction when the user clicks the button
if st.button("Predict"):
    input_data = [enrollees, hospital_reimbursements, physician_reimbursements, outpatien
                  home_health_reimbursements, hospice_reimbursements, durable_equipment_r

    # Validate input
    if not any(input_data):
        st.error("Please enter valid inputs for all fields.")
    else:
        prediction = predict_medicare_reimbursements(input_data)
        st.success(f"The predicted total Medicare reimbursements per enrollee is: ${predi

# Display model performance on the test set
st.write("Model Performance on Test Data:")
y_pred = model.predict(X_test)
r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
st.write(f"R-squared: {r2:.2f}")
st.write(f"Mean Squared Error: {mse:.2f}")


    Writing app.py
```

```
Writing app.py
```

Start coding or generate with AI.

```
# Write the Streamlit app code to a file
%%writefile app.py

import streamlit as st
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error
from sklearn.model_selection import train_test_split


st.markdown("""
    <style>
    .main {
        backgroundcolor:#4CAF50;
        background-size: cover;
        color: red;
    }
    .title {
        font-family: 'Arial', sans-serif;
        color: #FF5733;
        text-align: center;
        margin-top: 30px;
    }
    .text-input {
        background: rgba(255, 255, 255, 0.7);
        border-radius: 5px;
        padding: 10px;
        margin: 10px 0;
    }
    </style>
    """, unsafe_allow_html=True)

# Read the dataset from the uploaded CSV file

df = pd.read_csv('/content/drive/MyDrive/cleaned_data.csv')

# Display the first few rows of the dataframe for verification
st.write("Sample Data:")
st.write(df.head())

# Update columns according to your dataset
# Assuming your dataset has the following columns:
# 'Medicare enrollees (2014)', 'Hospital & skilled nursing facility reimbursements per en
# 'Physician reimbursements per enrollee (2014)', 'Outpatient facility reimbursements per
# 'Home health agency reimbursements per enrollee (2014)', 'Hospice reimbursements per en
```

```
# 'Home health agency reimbursements per enrollee (2014)', 'Hospice reimbursements per en
# 'Durable medical equipment reimbursements per enrollee (2014)', 'Total Medicare reimbur

# Splitting the dataset
X = df[['Medicare enrollees (2014)', 'Hospital & skilled nursing facility reimbursements
        'Physician reimbursements per enrollee (2014)', 'Outpatient facility reimbursemen
        'Home health agency reimbursements per enrollee (2014)', 'Hospice reimbursements
        'Durable medical equipment reimbursements per enrollee (2014)']]
y = df['Total Medicare reimbursements per enrollee (2014)']

# Splitting 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)

# Initializing and training the model
model = LinearRegression()
model.fit(X_train, y_train)

# Define a function for making predictions
def predict_medicare_reimbursements(input_data):
    prediction = model.predict(np.array(input_data).reshape(1, -1))
    return prediction[0]

# Streamlit app
st.title("Medicare Reimbursement Predictor")
st.write("Enter the details to predict the total Medicare reimbursements per enrollee.")

# User input for each feature
enrollees = st.number_input('Medicare enrollees (2014)', min_value=0)
hospital_reimbursements = st.number_input('Hospital & skilled nursing facility reimbursem
physician_reimbursements = st.number_input('Physician reimbursements per enrollee (2014)'
outpatient_reimbursements = st.number_input('Outpatient facility reimbursements per enrol
home_health_reimbursements = st.number_input('Home health agency reimbursements per enrol
hospice_reimbursements = st.number_input('Hospice reimbursements per enrollee (2014)', mi
durable_equipment_reimbursements = st.number_input('Durable medical equipment reimburseme

# Make a prediction when the user clicks the button
if st.button("Predict"):
    input_data = [enrollees, hospital_reimbursements, physician_reimbursements, outpatien
                  home_health_reimbursements, hospice_reimbursements, durable_equipment_r

    # Validate input
    if not any(input_data):
        st.error("Please enter valid inputs for all fields.")
    else:
        prediction = predict_medicare_reimbursements(input_data)
        st.success(f"The predicted total Medicare reimbursements per enrollee is: ${predi

# Display model performance on the test set
st.write("Model Performance on Test Data:")
y_pred = model.predict(X_test)
r2 = r2_score(y_test, y_pred)
```

```
mse = mean_squared_error(y_test, y_pred)
st.write(f"R-squared: {r2:.2f}")
st.write(f"Mean Squared Error: {mse:.2f}")
```

```
Overwriting app.py
```