Title: Medical Insurance Cost Prediction

By Harish K M

Abstract:

This project builds a machine learning system to predict individual medical insurance costs using the factors such as age, gender, BMI, smoking status, and dependents. The system involves data pre-processing, feature engineering, EDA, model training (Linear Regression, Random Forest, XGBoost, etc.), performance evaluation, and deployment using Streamlit. MLflow is used to track experiment runs and register the best-performing model. The project aims to support insurance companies and policyholders with accurate cost estimations and transparent pricing.

Introduction:

Medical insurance costs are a significant financial concern for individuals and families, especially in an era where healthcare expenses are rising globally. Insurance providers must determine premiums that accurately reflect each individual's risk profile, while consumers seek cost transparency and personalized policy planning. Traditional methods of estimating insurance charges often rely on fixed brackets or generalized assumptions, which may not account for unique combinations of demographic and lifestyle factors.

This project aims to develop a machine learning-based regression system to predict individual medical insurance charges using real-world data. The prediction is based on key variables such as age, gender, body mass index (BMI), smoking status, number of children, and region of residence. By leveraging machine learning techniques, the system provides data-driven, personalized cost predictions that can benefit both insurers and insured individuals.

The project follows an end-to-end data science workflow:

- **Data Preprocessing:** Cleaning and preparing the dataset, encoding categorical variables, and performing feature engineering.
- **Exploratory Data Analysis (EDA):** Visualizing and understanding how each factor affects insurance costs.
- **Model Development:** Training and evaluating multiple regression models (e.g., Linear Regression, Random Forest, XGBoost) using metrics such as RMSE, MAE, and R^2.
- **Experiment Tracking:** Using MLflow to track hyperparameters, model performance, and manage the model lifecycle.
- **Deployment:** Building an interactive Streamlit web app that allows users to input their personal data and get real-time predictions of their expected insurance charges.

This system supports multiple stakeholders:

- Insurance companies in pricing policies more accurately.

- Individuals in planning and comparing policies.
- Consultants and agents in providing personalized cost estimates.

By combining machine learning with an intuitive user interface, the project provides a practical tool for cost estimation in the healthcare insurance sector.

Dataset Description:

The dataset used in this project is a publicly available medical insurance dataset that includes key demographic and lifestyle attributes of individuals, along with their actual medical insurance charges. charges are the target variable representing the medical insurance cost billed to the individual. It is a continuous variable and will be predicted using regression models. This dataset is ideal for regression analysis and machine learning experimentation because it contains both numerical and categorical features. And the relationship between features and the target is nonlinear and interaction-driven, making it well-suited for advanced models like Random Forest or XGBoost.

Data Preprocessing:

Preprocessing is a crucial step in building robust machine learning models. It involves preparing the raw dataset by transforming categorical variables, creating meaningful features, and generating additional variables that capture deeper relationships among inputs.

This section outlines the preprocessing tasks applied to the medical insurance dataset and the rationale behind each transformation.

a) Categorical Variable Encoding:

The dataset includes three categorical columns: sex, smoker, and region. Since most machine learning models require numerical inputs, these variables must be encoded.

CODE:

```
label_encoders = {}

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

    le = LabelEncoder()

    df[col] = le.fit_transform(df[col])

    label_encoders[col] = le
```

Label Encoding converts each category into an integer (e.g., male is 1, female is 0). It allows the model to process these non-numeric features. Label encoders

are stored (`label_encoders[col] = le`) for future use, such as converting predictions back to readable labels.

b) Feature Engineering: BMI Category

To enhance interpretability and capture non-linear effects of BMI, a new categorical feature called bmi_category is created based on standard health ranges.

CODE:

```
def bmi_category(bmi):
    if bmi < 18.5: return 'underweight'
    elif 18.5 <= bmi < 25: return 'normal'
    elif 25 <= bmi < 30: return 'overweight'
    else: return 'obese'
df['bmi_category'] = df['bmi'].apply(bmi_category)
```

Classifying BMI helps the model recognize health risk levels. Categorical BMI groups may influence insurance charges differently compared to raw BMI values.

c) One-Hot Encoding: BMI Category

Since bmi_category is a categorical feature, we use one-hot encoding to convert it into binary columns.

CODE:

```
df = pd.get_dummies(df, columns=['bmi_category'], drop_first=True)
```

One-hot encoding avoids misinterpreting BMI categories as ordered values. `drop_first=True` prevents multicollinearity by dropping one base category.

d) Interaction Terms:

Interaction features are added to capture combined effects between two variables. For example, the impact of age might differ for smokers and non-smokers.

CODE:

```
df['age_smoker'] = df['age'] * df['smoker']
df['bmi_smoker'] = df['bmi'] * df['smoker']
```

```
df['age_bmi'] = df['age'] * df['bmi']
```

`age_smoker` captures how aging affects costs differently for smokers. `bmi_smoker` reflects how higher BMI combined with smoking could increase risks. `age_bmi` explores whether older individuals with higher BMI are at more risk. These new features help the model detect non-obvious patterns that individual features alone may not explain.

e) Saving the Cleaned Dataset:

After all transformations, the final cleaned and feature-enriched dataset is saved for further analysis and modelling:

CODE:

```
df.to_csv("medical insurance_cleaned.csv")
```

Exploratory Data Analysis (EDA):

Exploratory Data Analysis (EDA) is performed to understand the structure, distribution, and relationships within the dataset. It helps uncover insights that guide feature engineering and model selection. In this project, EDA focuses on identifying which features significantly influence medical insurance charges and how they interact.

a) Univariate Analysis (Single Variable):

What is the distribution of medical insurance charges?

CODE:

```
plt.figure(figsize=(8, 5))

sns.histplot(df['charges'], kde=True, bins=40,
color='maroon')

plt.title("Distribution of Medical Insurance Charges")

plt.xlabel("Charges")

plt.ylabel("Frequency")

plt.grid(True)

plt.tight_layout()

plt.show()
```

Interpretation:

The distribution of medical insurance charges is right-skewed (positively skewed). Most individuals pay charges concentrated between ₹5,000 and ₹15,000. However, there is a long tail extending toward higher values some people are paying over ₹50,000 or even ₹60,000.

The distribution of insurance charges is heavily skewed to the right, with the majority of individuals paying moderate premiums and a minority paying significantly high amounts. These high charges are typically linked to risk factors such as smoking, obesity, and advanced age.

What is the age distribution of the individuals?

CODE:

```
plt.figure(figsize=(8, 5))

sns.histplot(df['age'], kde=True, bins=20,
color='skyblue')

plt.title("Age Distribution of Individuals")

plt.xlabel("Age")

plt.ylabel("Count")

plt.tight_layout()

plt.show()
```

Interpretation:

The age variable in the dataset appears to be fairly uniformly distributed between 18 and 64 years. There are no noticeable spikes or gaps in the distribution—every age group has nearly equal representation. This balanced representation ensures that the trained model will learn from diverse age groups without age-related bias. Age is a crucial factor in medical cost estimation, as older individuals are generally more likely to incur higher medical expenses.

How many people are smokers vs non-smokers?

CODE:

```
plt.figure(figsize=(6, 4))

sns.countplot(x='smoker', hue='smoker', data=df,
palette='Set2', legend=False)
```

```
plt.title("Smokers vs Non-Smokers")

plt.xticks(ticks=[0, 1], labels=["No", "Yes"])

plt.ylabel("Count")

plt.tight_layout()

plt.show()
```

Interpretation:

The plot shows a significantly higher number of non-smokers than smokers in the dataset. This indicates that the majority of the population in the dataset does not smoke. Since smoking is a major factor influencing medical insurance charges, this imbalance should be considered when training the model. While this reflects real-world demographics, it also introduces a class imbalance that should be carefully considered during model training, especially since smoking has a strong influence on medical costs.

What is the average BMI in the dataset?

CODE:

```
mean_bmi = df['bmi'].mean()

print(f" Average BMI: {mean_bmi:.2f}")

plt.figure(figsize=(8, 5))

sns.histplot(df['bmi'], kde=True, color='mediumseagreen')

plt.axvline(mean_bmi, color='red', linestyle='--',
label=f"Mean BMI: {mean_bmi:.2f}")

plt.title("Distribution of BMI")

plt.xlabel("BMI")

plt.legend()

plt.tight_layout()

plt.show()
```

Interpretation:

The histogram shows a right-skewed distribution of BMI values. The distribution is slightly right-skewed, with a considerable number of individuals having high BMI values. This is an important insight, as BMI is directly linked to increased medical costs and chronic health risks. Such data patterns justify the use of BMI both as a raw feature and as a categorized risk factor during feature engineering. The average BMI is 30.66, which technically falls in the "obese" category (BMI > 30). The red dashed line represents the mean BMI, and a noticeable portion of the population lies to the right of this mean—indicating a significant number of individuals with obesity.

Which regions have the most number of policyholders?

CODE:

```
plt.figure(figsize=(6, 4))

sns.countplot(x='region', hue='region', data=df,
order=df['region'].value_counts().index,
palette='pastel', legend=False)

plt.title("Number of Policyholders by Region")

plt.xlabel("Region")

plt.ylabel("Count")

plt.xticks(ticks=[0, 1, 2, 3], labels=["NorthEast",
"NorthWest", "SouthEast", "SouthWest"])

plt.tight_layout()

plt.show()
```

Interpretation:

The NorthEast region has the highest number of policyholders, followed closely by the other three. This distribution suggests geographic diversity in the dataset, which is valuable for generalizing model predictions.

b) Bivariate Analysis (Two Variables):

How do charges vary with age?

CODE:

```
plt.figure(figsize=(8, 5))
```

```
sns.regplot(x='age', y='charges', data=df,
scatter_kws={'alpha':0.4}, line_kws={'color':'red'})

plt.title("Charges vs Age")

plt.xlabel("Age")

plt.ylabel("Medical Charges")

plt.tight_layout()

plt.show()
```

Interpretation:

The scatter plot shows a positive correlation between age and medical charges. As individuals age, healthcare needs and risks generally increase, leading to higher insurance costs. The red regression line highlights this upward trend, although the variability in charges also increases with age. The trendline confirms this linear relationship, although the spread of charges becomes wider at older ages, indicating greater variability due to additional factors like BMI and smoking status. There are some sharp upward spikes in charges for older individuals likely due to high-risk individuals or smokers.

Is there a difference in average charges between smokers and non-smokers?

CODE:

```
plt.figure(figsize=(6, 4))

sns.boxplot(x='smoker', y='charges', data=df,
hue='smoker', palette='Set2', legend=False)

plt.xticks([0, 1], ['Non-Smoker', 'Smoker'])

plt.title("Medical Charges: Smokers vs Non-Smokers")

plt.tight_layout()

plt.show()
```

Interpretation:

The boxplot clearly shows that smokers pay significantly higher medical charges than non-smokers. Smokers not only have higher median charges, but the range and number of high-cost outliers are also much greater. Non-smokers typically have lower charges with a tighter distribution. This strong separation validates the

inclusion of smoking status as a critical feature in the predictive model, and supports insurance pricing strategies that account for lifestyle-related risk factors.

Does BMI impact insurance charges?

CODE:

```
plt.figure(figsize=(8, 5))

sns.regplot(x='bmi', y='charges', data=df,
scatter_kws={'alpha':0.3}, line_kws={'color':'green'})

plt.title("Charges vs BMI")

plt.xlabel("BMI")

plt.ylabel("Charges")

plt.tight_layout()

plt.show()
```

Interpretation:

The scatter plot suggests a weak positive correlation between BMI and insurance charges. While charges tend to rise slightly with higher BMI, the relationship is not as strong or linear as with age or smoking status. A few individuals with very high charges and high BMI may be influencing the regression line.

Do men or women pay more on average?

CODE:

```
plt.figure(figsize=(6, 4))

sns.boxplot(x='sex', y='charges', data=df, hue='sex',
palette='pastel', legend=False)

plt.xticks([0, 1], ['Female', 'Male'])

plt.title("Medical Charges: Male vs Female")

plt.tight_layout()

plt.show()
```

```
gender_charges = df.groupby('sex')['charges'].mean()

abs_diff = abs(gender_charges[1] - gender_charges[0])

print(f"Average difference in charges (Male - Female):
₹{abs_diff:.2f}")

from scipy.stats import ttest_ind

male_charges = df[df['sex'] == 1]['charges']

female_charges = df[df['sex'] == 0]['charges']

t_stat, p_value = ttest_ind(male_charges, female_charges)

print(f"T-statistic: {t_stat:.3f}, p-value:
{p_value:.3f}")
```

Interpretation:

If p-value > 0.05: No significant difference means the difference in average charges between men and women is statistically significant. If p-value <= 0.05: Statistically significant difference. You can reject the null hypothesis that "both genders have the same mean charges". Men and women show a statistically significant difference in average medical charges (p = 0.001), with men paying about ₹1,500 more on average.

Is there a correlation between number of children and charges?

CODE:

```
plt.figure(figsize=(7, 4))

sns.violinplot(x='children', y='charges', hue='children',
data=df, palette='coolwarm', legend=False )

plt.title("Distribution of Charges by Children Count")

plt.tight_layout()

plt.show()

corr = df['children'].corr(df['charges'])

print(f"Correlation between number of children and
charges: {corr:.4f}")
```

Interpretation:

The distribution of charges remains fairly consistent across different values of `children` (0 to 5). All categories exhibit right-skewed distributions, likely due to the presence of high-cost outliers. The correlation coefficient value suggests a very weak positive linear relationship, which is not statistically meaningful.

c) Multivariate Analysis (More than Two Variables):

How does smoking status combined with age affect medical charges?

CODE:

```
plt.figure(figsize=(8, 5))

sns.scatterplot(x='age', y='charges', hue='smoker',
data=df, palette='Set1', alpha=0.6)

plt.title("Charges vs Age by Smoking Status")

plt.xlabel("Age")

plt.ylabel("Medical Charges")

plt.legend(title="Smoker (1=Yes, 0=No)")

plt.tight_layout()

plt.show()
df.groupby('smoker')['charges'].mean()
```

Interpretation:

There is a clear separation between smokers and non-smokers: Smokers (1) are clustered much higher on the y-axis, indicating higher charges at nearly all ages. Non-smokers (0) show a relatively flatter and lower spread of charges. As age increases, charges for smokers rise much more steeply than for non-smokers. This reflects the compounding risk that smoking + aging brings to insurance costs. Smokers who have significantly higher charges, especially as age increases. The gap widens in the 35+ age group.

What is the impact of gender and region on charges for smokers?

CODE:

```
smokers_df = df[df['smoker'] == 1]

plt.figure(figsize=(10, 5))

sns.boxplot(x='region', y='charges', hue='sex',
data=smokers_df, palette='Set2')
```

```
plt.title("Medical Charges for Smokers by Region and
Gender")

plt.xlabel("Region Code (0-Northeast, 1-Northwest, 2-
Southeast, 3-Southwest)")

plt.ylabel("Charges")

plt.legend(title="Sex (0=Female, 1=Male)")

plt.tight_layout()

plt.show()
smokers_df.groupby(['region', 'sex'])['charges'].mean()
```

Interpretation:

This bivariate analysis reveals that among smokers, both region and gender significantly influence medical charges. The box plot shows noticeable variations in median and spread of charges across the four regions. Male smokers in southeast and northeast often show higher charges. Gender difference appears region-dependent.

How do age, BMI, and smoking status together affect insurance cost?

CODE:

```
plt.figure(figsize=(9, 6))

sns.scatterplot(x='age', y='bmi', size='charges',
hue='smoker', data=df, sizes=(20, 300), alpha=0.6,
palette='coolwarm')

plt.title("Age vs BMI Colored by Smoker and Sized by
Charges")

plt.tight_layout()

plt.show()
```

Interpretation:

Larger bubbles (higher charges) are mostly seen among: Older individuals, With higher BMI and Who are smokers (colored in red shades). Non-smokers (blue colors) tend to have smaller bubbles across all age and BMI values, indicating lower charges on average. Among younger individuals (age < 30), even with high BMI, the charges remain moderate unless they are smokers.

Do obese smokers (BMI > 30) pay significantly higher than non-obese non-smokers?

CODE:

```
df['obese'] = (df['bmi'] > 30).astype(int)

group1 = df[(df['smoker'] == 1) & (df['obese'] == 1)]  #
Obese Smokers

group2 = df[(df['smoker'] == 0) & (df['obese'] == 0)]  #
Non-obese Non-Smokers

mean1 = group1['charges'].mean()

mean2 = group2['charges'].mean()

diff = mean1 - mean2

print(f"Obese Smokers Avg: ₹{mean1:.2f}")

print(f"Non-obese Non-Smokers Avg: ₹{mean2:.2f}")

print(f"Difference: ₹{diff:.2f}")

df['group'] = 'Other'

df.loc[(df['smoker'] == 1) & (df['obese'] == 1), 'group']
= 'Obese Smoker'

df.loc[(df['smoker'] == 0) & (df['obese'] == 0), 'group']
= 'Non-Obese Non-Smoker'

plt.figure(figsize=(6, 4))

sns.boxplot(x='group', y='charges', hue='group',
data=df[df['group'] != 'Other'], palette='Set3')

plt.title("Charges: Obese Smokers vs Non-Obese Non-
Smokers")

plt.tight_layout()

plt.show()

from scipy.stats import ttest_ind

t_stat, p_val = ttest_ind(group1['charges'],
group2['charges'], equal_var=False)

print(f"T-statistic: {t_stat:.2f}, p-value: {p_val:.4f}")
```

Interpretation:

A p-value < 0.05 means the result is statistically significant. p = 0.0000 means the likelihood this difference happened by chance is less than 0.01%. The two groups

have significantly different average insurance charges. Obese smokers pay statistically significantly higher medical insurance charges than non-obese non-smokers. This suggests BMI and smoking together greatly increase healthcare costs — and are important factors for insurers.

d) Outlier Detection:

Are there outliers in the charges column? Who are the individuals paying the highest costs?

CODE:

```
plt.figure(figsize=(8, 4))

sns.boxplot(x=df['charges'], color='salmon')

plt.title("Boxplot of Medical Charges")

plt.xlabel("Charges")

plt.tight_layout()

plt.show()

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

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

IQR = Q3 - Q1

lower = Q1 - 1.5 * IQR

upper = Q3 + 1.5 * IQR

outliers = df[df['charges'] > upper]

print(f"Outliers in charges: {len(outliers)} records")

top5 = df.sort_values(by='charges',
ascending=False).head(5)

print(top5[['age', 'sex', 'bmi', 'children', 'smoker',
'region', 'charges']])
```

Interpretation:

These aren't wrong — they likely represent real high-risk individuals (e.g., older smokers with obesity). Medical charges often follow a right-skewed distribution, where a small group pays disproportionately high amounts.

Are there extreme BMI values that could skew predictions?

CODE:

```
plt.figure(figsize=(7, 4))

sns.boxplot(x=df['bmi'], color='skyblue')

plt.title("Boxplot of BMI")

plt.xlabel("BMI")

plt.tight_layout()

plt.show()

Q1_bmi = df['bmi'].quantile(0.25)

Q3_bmi = df['bmi'].quantile(0.75)

IQR_bmi = Q3_bmi - Q1_bmi

lower_bmi = Q1_bmi - 1.5 * IQR_bmi

upper_bmi = Q3_bmi + 1.5 * IQR_bmi

bmi_outliers = df[df['bmi'] > upper_bmi]

print(f" Extreme BMI outliers: {len(bmi_outliers)}
individuals")
```

Interpretation:

Medical charges show many high-end outliers — likely due to smoking and age.

e) Correlation Analysis:

What is the correlation between numeric features like age, BMI, number of children, and charges?

CODE:

```
numeric_cols = ['age', 'bmi', 'children', 'charges']
corr_matrix = df[numeric_cols].corr()
print("Correlation Matrix:")
print(corr_matrix)
```

Which features have the strongest correlation with the target variable (charges)?

CODE:

```
plt.figure(figsize=(6, 4))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm',
fmt=".2f")
```

```
plt.title("Correlation Heatmap: Numeric Features vs
Charges")
plt.tight_layout()
plt.show()
```

Interpretation:
Age has the moderate positive correlation with charges among numeric features. Means older individuals tend to pay more.

Model Building:

a) **Setup and Preprocessing:**
Libraries:
CODE:

```
import pandas as pd
import numpy as np
import mlflow
import mlflow.sklearn
import xgboost as xgb
import joblib

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Ridge,
Lasso
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor,
GradientBoostingRegressor
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error,
mean_absolute_error, r2_score
from sklearn.preprocessing import StandardScaler
```

MLflow Setup:
CODE:

```
mlflow.set_tracking_uri("http://127.0.0.1:5000/")
mlflow.set_experiment("Medical_Insurance_Cost_Prediction"
)
```

Why mlflow?

MLflow is a machine learning lifecycle management tool. It helps track, organize, reproduce, and deploy ML models and experiments.

Mlflow solves:
Tracks all runs, metrics, and parameters. Automatically saves code, hyperparameters, environment, and data path. Dashboard view of all experiments. Supports saving and serving models via API. Centralized experiment tracking for all users.

Data Split:

Features X include age, sex, bmi, children, smoker, region; the target is charges. The we split the data for train and test.

CODE:

```
X = df[['age', 'sex', 'bmi', 'children', 'smoker',
'region']]
y = df['charges']

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

Scaling:

Applies StandardScaler to standardize features for models sensitive to scale.(eg Linear Regression, Ridge, Lasso and SVR)

CODE:

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

b) Model:

a) LinearRegression:
Predicts charges using a linear combination of input features. A good baseline; interpretable and fast.

b) Ridge:
Linear regression with L2 regularization. Controls overfitting, especially with multicollinearity.

c) Lasso:
Linear regression with L1 regularization. Also controls overfitting, and performs **feature selection** by shrinking less important feature coefficients to zero.

d) Decision Tree Regressor:
Non-linear model that splits data into decision rules. Captures non-linear interactions, works well without scaling.

e) Random Forest Regressor:
Ensemble of decision trees using bagging. Reduces overfitting, handles feature importance, robust to outliers.

f) XGBoost Regressor:
Gradient boosting model, more powerful than random forest. Often delivers state-of-the-art results in structured data. Handles missing data, high bias-variance tradeoff control.

g) SVR (Support Vector Regressor):
Fits the best boundary within a margin (ε-insensitive). Effective for high-dimensional data, robust to outliers, but requires scaling.

h) Gradient Boosting Regressor:

Similar to XGBoost, but pure sklearn version. Handles bias well, builds models sequentially to improve accuracy.

CODE:

```
models = {
    "LinearRegression": LinearRegression(),
    "Ridge": Ridge(alpha=1.0),
    "Lasso": Lasso(alpha=0.01),
    "DecisionTree": DecisionTreeRegressor(max_depth=5),
    "RandomForest":
RandomForestRegressor(n_estimators=100,
random_state=42),
    "XGBoost": xgb.XGBRegressor(n_estimators=100,
max_depth=4, learning_rate=0.1, random_state=42),
    "SVR": SVR(kernel='rbf', C=100, epsilon=0.1),
    "GradientBoosting":
GradientBoostingRegressor(n_estimators=100,
learning_rate=0.1, max_depth=4, random_state=42)
}
```

c) Metrics:

Each model is trained (scaled where needed) and evaluated using:

- **RMSE (Root Mean Squared Error)** – penalizes large errors more heavily.
- **MAE (Mean Absolute Error)** – gives an average of absolute errors.
- **R^2 (R-squared)** – percentage of variance in the target explained by the model.

CODE:

```
rmse = np.sqrt(mean_squared_error(y_test,
preds))

mae = mean_absolute_error(y_test, preds)

r2 = r2_score(y_test, preds)

mlflow.log_param("model_name", name)

mlflow.log_metric("RMSE", rmse)

mlflow.log_metric("MAE", mae)

mlflow.log_metric("R2", r2)

mlflow.sklearn.log_model(model, "model")
```

d) Best Model Registration:

The best model is saved to MLflow under a registered name "BestInsuranceModel". Also saved locally as a .pkl file using joblib.

CODE:

```
if rmse < best_score:

          best_score = rmse

          best_model = model

          best_model_name = name

     with
mlflow.start_run(run_name="Register_Best_Model") as
run:

mlflow.sklearn.log_model(best_model, "best_model",
registered_model_name="BestInsuranceModel")

mlflow.log_param("best_model", best_model_name)

mlflow.log_metric("best_RMSE", best_score)

joblib.dump(best_model, "best_model.pkl")
```

Streamlit:

Allows for quick and easy web app development in pure Python. Ideal for data science and machine learning projects. Supports seamless integration of plots (Matplotlib, Seaborn) and interactive widgets. Supports loading pre-trained models and integrating prediction logic directly into the UI.

App Structure:

The Streamlit application was divided into several logical sections using a sidebar menu:

1. **Home:**
   - Brief introduction to the dataset and project.
   - Purpose of the application.
2. **Univariate Analysis:**
   - Explored the distribution of individual variables (age, bmi, charges, etc.).

- Visualizations: Histograms and KDE plots to assess skewness, range, and central tendency.

3. **Bivariate Analysis:**

- Analyzed relationships between two variables, such as:
  - bmi vs charges
  - smoker vs charges
  - sex vs charges
- Visualizations: Scatter plots, box plots, and bar plots.

4. **Multivariate Analysis:**

- Explored interactions between three or more variables.
- Example: charges by region, sex, and smoker.
- Used grouped bar plots and multivariate box plots.

5. **Outlier Detection:**

- Identified extreme values in bmi and charges using boxplots and IQR method.
- Displayed count of potential outliers to assess impact on prediction accuracy.

6. **Correlation:**

- Showed correlation matrix among numeric variables: age, bmi, children, charges.
- Visualization: Heatmap.

7. **Prediction:**

- Loaded the best trained ML model (best_model.pkl) using joblib.
- Collected user inputs for features (age, sex, bmi, etc.) using st.number_input() and st.selectbox().
- Scaled input if needed, then displayed the predicted insurance charge.
- Implemented input validation with error handling.

The Streamlit app provided an easy-to-use and accessible platform for users to explore the medical insurance dataset and predict charges interactively. It served both as a tool for analysis and decision support by showing how lifestyle and demographic factors impact insurance costs.

Business Use Cases:

1. **Insurance Companies – Personalized Pricing**

- **Objective**: Offer data-driven, personalized policy pricing to clients.
- **How the App Helps**:

- Insurers can input a customer's demographic and health information (e.g., age, BMI, smoking status) to estimate fair premium **charges** using the trained model.
- It enables risk-based pricing, allowing companies to align premiums with potential healthcare costs.
- Identifies high-risk individuals (e.g., smokers with high BMI), enabling preventive outreach or tailored policies.

2. **Individuals – Cost Comparison Before Choosing a Policy**

- **Objective**: Empower customers with estimated insurance costs before purchasing.
- **How the App Helps**:

  - Users can enter their information and receive a realistic charge estimate based on their lifestyle and health profile.
  - They can compare scenarios - for example, understanding how quitting smoking or reducing BMI could lower insurance costs.
  - Helps in budget planning and choosing the best policy based on predicted charges.

3. **Financial Advisors - Estimate Medical Costs:**
   - **Objective**: Help clients plan future financial needs and insurance coverage.
   - **How the App Helps**:

     - Advisors can simulate various life situations (aging, childbirth, weight gain/loss) and estimate how these affect insurance premiums.
     - Offers a financial planning tool to assess long-term health insurance needs and coverage gaps.
     - Can recommend preventive steps based on potential cost savings.

4. **General Public - Transparency in Pricing:**
   - **Objective**: Increase awareness of how insurance pricing works.
   - **How the App Helps**:

     - Makes the pricing model transparent and understandable by linking costs with key personal attributes.
     - Educates users on how behaviors like smoking or obesity influence premiums.
     - Encourages healthier lifestyles by showing financial benefits of good health.

Conclusion:

In this project, we successfully developed a robust medical insurance cost prediction system by combining machine learning, experiment tracking, and user interface design. Below is a summary of key achievements and insights:

Key Outcomes:

- **Built a functional prediction pipeline**
  A working regression system was created to estimate individual medical insurance charges using key demographic and lifestyle features (age, BMI, smoker status, etc.).
- **Trained and evaluated multiple models**
  Explored several regression models including Linear Regression, Random Forest, XGBoost, and Gradient Boosting. XGBoost yielded the best performance based on RMSE and R^2 metrics.
- **Used MLflow for experiment tracking**
  Enabled streamlined model comparison and logging of parameters, metrics, and artifacts. This ensured reproducibility and transparency in experimentation.
- **Deployed a user-friendly Streamlit app**
  Designed a lightweight web app that allows end-users to input their details and receive an estimated insurance charge instantly, enhancing usability and real-world application.
- **Model Optimization**
  Apply hyperparameter tuning (e.g., GridSearchCV) to further improve model accuracy and reduce overfitting.
- **Feature Engineering**
  Include more relevant health-related features like exercise frequency, diet, medical history, etc.
- **Explainability Tools**
  Integrate SHAP or LIME to improve model interpretability for stakeholders.
- **Deployment Scaling**
  Host the app on cloud platforms (e.g., AWS, Heroku) for broader accessibility and scalability.