## **🧠 Project Breakdown**

### **1. Importing Required Libraries**

import pandas as pd  
import seaborn as sns  
import matplotlib.pyplot as plt  
import statsmodels.api as sm  
from sklearn.model\_selection import train\_test\_split  
from sklearn.linear\_model import LinearRegression  
from sklearn.metrics import mean\_squared\_error, r2\_score

These libraries help with:

* **Data handling** (pandas)
* **Data visualization** (seaborn, matplotlib)
* **Statistical modeling** (statsmodels, sklearn)
* **Model evaluation metrics** (mean\_squared\_error, r2\_score)

### **2. Loading the Data**

dataframe1 = pd.read\_csv("Medicalpremium.csv")

This loads the medical insurance premium dataset into a DataFrame. Typically, the dataset contains:

* **Independent variables** (age, BMI, smoking status, etc.)
* **Dependent variable**: PremiumPrice (insurance cost)

### **3. Statistical Summary**

print(dataframe1.describe())

🔍 **Statistical Concept: Descriptive Statistics**

This shows:

* **Mean**: Central tendency
* **Standard Deviation**: Spread of data
* **Min/Max, Quartiles**: Distribution insights

Useful to understand data ranges, check for outliers, and prepare for normalization or transformation if needed.

### **4. Correlation Matrix**

sns.heatmap(dataframe1.corr(), annot=True, cmap="coolwarm")

🔍 **Statistical Concept: Pearson Correlation**

* Measures **linear relationship** between numeric variables
* Range: -1 to 1
  + 1: perfect positive correlation
  + -1: perfect negative correlation
  + 0: no correlation

Helps identify **multicollinearity** and **feature relevance** for prediction.

### **5. Feature & Target Splitting**

x\_data = dataframe1.drop("PremiumPrice", axis=1)  
y\_data = dataframe1["PremiumPrice"]

* x\_data: All independent variables (features)
* y\_data: Dependent variable (PremiumPrice)

### **6. Train-Test Split**

x\_train, x\_test, y\_train, y\_test = train\_test\_split(...)

🔍 **Statistical Concept: Model Validation**

* Splits data into **training (70%)** and **testing (30%)**.
* Ensures model is evaluated on **unseen data**, avoiding overfitting.

### **7. Linear Regression (sklearn)**

model = LinearRegression()  
model.fit(x\_train, y\_train)  
y\_pred = model.predict(x\_test)

🔍 **Statistical Concept: Linear Regression**

* **Multivariate Linear Regression** predicts y using **multiple x variables**.
* Finds the best-fit line:

*y=β0+β1x1+β2x2+⋯+βnxn+ϵy = \beta\_0 + \beta\_1x\_1 + \beta\_2x\_2 + \dots + \beta\_nx\_n + \epsilon*

Good for understanding how variables affect the outcome (e.g., how smoking status, age, BMI affect PremiumPrice).

### **8. Model Evaluation**

mean\_squared\_error(y\_test, y\_pred)  
r2\_score(y\_test, y\_pred)

* **MSE**: Mean Squared Error (average of squared prediction errors)

*MSE=1n∑(yi−y^i)2\text{MSE} = \frac{1}{n} \sum (y\_i - \hat{y}\_i)^2*

* **R² Score**: Coefficient of determination
  + Tells how well model explains variability in the target variable.
  + Ranges from 0 to 1 (higher is better)

### **9. OLS Regression Summary (statsmodels)**

x\_train\_sm = sm.add\_constant(x\_train)  
ols\_model = sm.OLS(y\_train, x\_train\_sm).fit()  
print(ols\_model.summary())

🔍 **Statistical Concept: Inferential Statistics**

This gives you:

* **Coefficients**: Effect of each feature on target
* **P-values**: Tests null hypothesis that coefficient = 0
  + **Low p-value (< 0.05)** → statistically significant
* **R-squared / Adjusted R-squared**
* **F-statistic**: Significance of the whole model
* **Confidence Intervals**

This is crucial for **hypothesis testing** and **model interpretability**.

### **10. Residual Plot**

residuals = y\_test - y\_pred  
sns.residplot(x=y\_pred, y=residuals, lowess=True)

🔍 **Statistical Concept: Assumption Checking**

Residuals help check if:

* Residuals are **randomly scattered** (good)
* Mean of residuals ≈ 0
* No clear pattern (→ linearity and homoscedasticity hold)

A non-random pattern indicates model issues like non-linearity or heteroscedasticity.