Healthcare Insurance Analysis - Project Solution

Step 1: Load and Explore the Data

1.1 Import Necessary Libraries

Ensure you have the required libraries installed. If not, install them using pip install pandas numpy matplotlib seaborn scikit-learn xgboost.
Now, import the libraries:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler, OneHotEncoder

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline

from sklearn.linear_model import LinearRegression

from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor

from xgboost import XGBRegressor

from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

Step 2: Load the Dataset

Replace 'insurance.csv' with your actual dataset file name.

```
df = pd.read_csv('insurance.csv')
```

Check the first few rows:

```
df.head()
```

Check dataset info:

```
df.info()
```

Check missing values:

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

Step 3: Exploratory Data Analysis (EDA)

3.1 Summary Statistics

```
df.describe()
```

3.2 Visualizing Data Distribution

Check distribution of charges (healthcare cost):

```
sns.histplot(df['charges'], kde=True)
plt.title('Distribution of Healthcare Charges')
plt.show()
```

3.3 Correlation Analysis

Check how numerical features correlate with healthcare costs:

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

3.4 Boxplots for Outliers

Check for outliers in charges based on different factors:

```
plt.figure(figsize=(12, 6))
sns.boxplot(x='smoker', y='charges', data=df)
plt.title('Effect of Smoking on Healthcare Charges')
plt.show()
```

```
plt.figure(figsize=(12, 6))
sns.boxplot(x='region', y='charges', data=df)
plt.title('Effect of Region on Healthcare Charges')
plt.show()
```

Step 4: Data Preprocessing

4.1 Handling Categorical Features

Encode categorical variables using one-hot encoding:

```
df = pd.get_dummies(df, columns=['sex', 'region', 'smoker'], drop_first=True)
```

4.2 Feature Scaling

Normalize numerical features:

```
scaler = StandardScaler()
df[['age', 'bmi', 'children']] = scaler.fit_transform(df[['age', 'bmi', 'children']])
```

4.3 Splitting Data

```
X = df.drop('charges', axis=1)
y = df['charges']

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

Step 5: Model Training & Evaluation

5.1 Train Models

Define models and evaluate them:

```
models = {
    "Linear Regression": LinearRegression(),
    "Decision Tree": DecisionTreeRegressor(),
    "Random Forest": RandomForestRegressor(),
    "Gradient Boosting": GradientBoostingRegressor(),
    "XGBoost": XGBRegressor()
}

results = {}

for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    results[name] = {
        "MAE": mean_absolute_error(y_test, y_pred),
        "RMSE": np.sqrt(mean_squared_error(y_test, y_pred)),
        "R2 Score": r2_score(y_test, y_pred)
    }

# Convert results to a DataFrame
results_df = pd.DataFrame(results).T
print(results_df)
```

Step 6: Insights & Recommendations

- Identify the best-performing model based on **RMSE and R^2 Score**.
- Check feature importance for tree-based models:

```
best_model = RandomForestRegressor()
best_model.fit(X_train, y_train)
```

```
feature_importances = pd.Series(best_model.feature_importances_, index=X.columns).sort_values(ascending=False)

plt.figure(figsize=(10, 6))

sns.barplot(x=feature_importances, y=feature_importances.index)

plt.title('Feature Importance')

plt.show()
```

Conclusion

1. **Which model performed best?** → Based on RMSE and R^2 Score.
2. **Key influencing factors** → Smoking, BMI, and Age likely have the highest impact.
3. **How insurance companies can use this** → Adjust premiums based on risk factors.