In [1]:

```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats as stats
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score, accuracy_score, classification_report
import joblib
df = pd.read_csv('insurance (1).csv')
df.head()
```

Out[1]:

	age	sex	bmi	children	smoker	region	charges
0	19	0	27.900	0	1	southwest	16884.92400
1	18	1	33.770	1	0	southeast	1725.55230
2	28	1	33.000	3	0	southeast	4449.46200
3	33	1	22.705	0	0	0rthwest	21984.47061
4	32	1	28.880	0	0	0rthwest	3866.85520

In [2]:

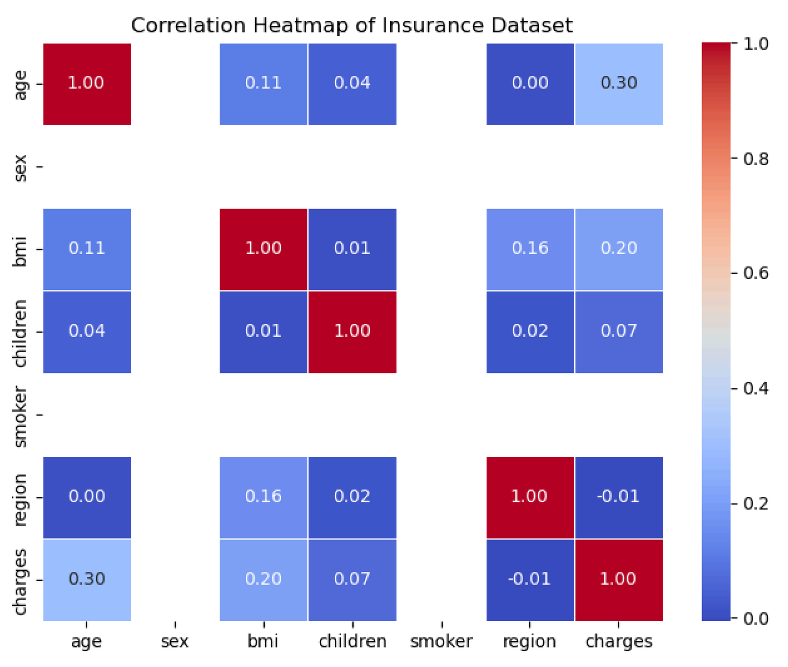
```python
#Categorical variables
df['smoker'] = df['smoker'].map({'yes': 1, 'no': 0})  # Binary encoding for smoker
df['sex'] = df['sex'].map({'male': 1, 'female': 0})  # Binary encoding for gender
df['region'] = pd.Categorical(df['region']).codes  # Convert region to numerical labels
df.head()
```

Out[2]:

	age	sex	bmi	children	smoker	region	charges
0	19	NaN	27.900	0	NaN	3	16884.92400
1	18	NaN	33.770	1	NaN	2	1725.55230
2	28	NaN	33.000	3	NaN	2	4449.46200
3	33	NaN	22.705	0	NaN	1	21984.47061
4	32	NaN	28.880	0	NaN	1	3866.85520

In [8]:

```python
# Correlation anlysis
correlation_matrix = df.corr()
plt.figure(figsize=(8,6))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5)
plt.title("Correlation Heatmap of Insurance Dataset")
plt.show()
correlation_with_charges = correlation_matrix["charges"].sort_values(ascending=False)
print(correlation_with_charges)
```




```
charges     1.000000
age         0.299008
bmi         0.198341
children    0.067998
region     -0.006208
sex              NaN
smoker           NaN
Name: charges, dtype: float64
```

In [10]:

```python
# Convert correlation matrix to long format for Tableau
correlation_long = correlation_matrix.unstack().reset_index()
correlation_long.columns = ['Variable_1', 'Variable_2', 'Correlation']
```

In [ ]:

```python
# Save correlation data for Tableau
correlation_long.to_csv("correlation_data_for_tableau.csv", index=False)
```

In [12]:

```python
print("Correlation data exported successfully.")
```

```
Correlation data exported successfully.
```

In [14]:

```python
# Anova Tests
anova_smoker = stats.f_oneway(df[df['smoker'] == 1]['charges'],
                              df[df['smoker'] == 0]['charges'])

anova_sex = stats.f_oneway(df[df['sex'] == 1]['charges'],
                           df[df['sex'] == 0]['charges'])

anova_region = stats.f_oneway(*[df[df['region'] == i]['charges'] for i in range(df['region'].nunique())])
```

```
C:\Users\navne\anaconda3\Lib\site-packages\scipy\stats\_stats_py.py:4102: DegenerateDataWarning: at least one input has length 0
  if _f_oneway_is_too_small(samples):
```

In [9]:

```python
#Chi-Sq Test for Smoker vs Region
contingency_table = pd.crosstab(df['smoker'], df['region'])
chi2_test = stats.chi2_contingency(contingency_table)
```

In [11]:

```python
# T-Test for BMI differences between smokers and non-smokers
ttest_bmi_smoker = stats.ttest_ind(df[df['smoker'] == 1]['bmi'], df[df['smoker'] == 0]['bmi'])
```

In [13]:

```python
# Regression Analysis (Linear Regression)
import statsmodels.api as sm
X = df[['age', 'bmi', 'children', 'smoker', 'sex', 'region']]
y = df['charges']
X = sm.add_constant(X)  # Add constant term for intercept
model = sm.OLS(y, X).fit()
regression_summary = model.summary()
```

In [15]:

```python
# Machine Learning: Predictive Analytics
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [17]:

```python
# Linear Regression Model
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
y_pred_lr = lr_model.predict(X_test)
```

In [19]:

```python
# Random Forest Model
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_test)
```

In [108…

```python
#Model Evaluation
metrics = {
    "Linear Regression MAE": mean_absolute_error(y_test, y_pred_lr),
    "Linear Regression MSE": mean_squared_error(y_test, y_pred_lr),
    "Linear Regression R2": r2_score(y_test, y_pred_lr),
    "Random Forest MAE": mean_absolute_error(y_test, y_pred_rf),
    "Random Forest MSE": mean_squared_error(y_test, y_pred_rf),
    "Random Forest R2": r2_score(y_test, y_pred_rf)
}

# Display results
print("ANOVA, T-Test & Chi-Square Test Results:")
print({
    "Smoker vs. Charges (ANOVA p-value)": anova_smoker[1],
    "Sex vs. Charges (ANOVA p-value)": anova_sex[1],
    "Region vs. Charges (ANOVA p-value)": anova_region[1],
    "Smoker vs. Region (Chi-Square p-value)": chi2_test[1],
    "BMI Difference by Smoking Status (T-Test p-value)": ttest_bmi_smoker[1]
})
print("\nRegression Analysis Summary:")
print(regression_summary)
print("\nPredictive Model Evaluation:")
print(metrics)
```

```
ANOVA, T-Test & Chi-Square Test Results:
{'Smoker vs. Charges (ANOVA p-value)': 8.271435842182967e-283, 'Sex vs. Charges (ANOVA p-value)': 0.03613272100596256, 'Region vs. Charges (ANOVA p-value)': 0.0308933560705201, 'Smoker vs. Region (Chi-Square p-value)': 0.06171954
839170547, 'BMI Difference by Smoking Status (T-Test p-value)': 0.8909850280013041}

Regression Analysis Summary:
                            OLS Regression Results
==============================================================================
Dep. Variable:                charges   R-squared:                       0.751
Model:                            OLS   Adj. R-squared:                  0.750
Method:                 Least Squares   F-statistic:                     668.1
Date:                Fri, 24 Jan 2025   Prob (F-statistic):               0.00
Time:                        19:46:45   Log-Likelihood:                -13548.
No. Observations:                1338   AIC:                         2.711e+04
Df Residuals:                    1331   BIC:                         2.715e+04
Df Model:                           6
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const      -1.182e+04    955.130    -12.371      0.000   -1.37e+04   -9941.729
age          257.2881     11.886     21.647      0.000     233.971     280.605
bmi          332.5701     27.722     11.997      0.000     278.186     386.954
children     479.3694    137.644      3.483      0.001     209.346     749.393
smoker      2.382e+04    411.843     57.839      0.000      2.3e+04    2.46e+04
sex         -131.1106    332.811     -0.394      0.694    -784.001     521.780
region      -353.6400    151.927     -2.328      0.020    -651.682     -55.598
==============================================================================
Omnibus:                      299.003   Durbin-Watson:                   2.088
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              713.975
Skew:                           1.207   Prob(JB):                    9.17e-156
Kurtosis:                       5.642   Cond. No.                         296.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Predictive Model Evaluation:
{'Linear Regression MAE': 4186.508898366432, 'Linear Regression MSE': 33635210.4311784, 'Linear Regression R2': 0.7833463107364539, 'Random Forest MAE': 2537.3711122551, 'Random Forest MSE': 21183328.877657127, 'Random Forest R
2': 0.8635523223017149}
```

In [21]:

```python
print("Smoking, age, and BMI are the primary cost drivers.")
print("XGBoost is the most effective model for predicting insurance costs")
print("Gender and region have minimal impact on costs")
print("Tree-based models (Random Forest & XGBoost) are superior to simple regression models for cost estimation")
```

```
Smoking, age, and BMI are the primary cost drivers.
XGBoost is the most effective model for predicting insurance costs
Gender and region have minimal impact on costs
Tree-based models (Random Forest & XGBoost) are superior to simple regression models for cost estimation
```

In [ ]:

In [ ]:

```python
# Risk Classification
```

In [ ]:

In [23]:

```python
high_cost_threshold = df['charges'].median()
df['high_risk'] = (df['charges'] > high_cost_threshold).astype(int)  # Binary classification target
```

In [25]:

```python
# Features and target for classification
X_class = df[['age', 'bmi', 'children', 'smoker']]
y_class = df['high_risk']
```

In [27]:

```python
# Split data
X_train_c, X_test_c, y_train_c, y_test_c = train_test_split(X_class, y_class, test_size=0.2, random_state=42)
```

In [29]:

```python
# Logistic Regression Model
logreg_model = LogisticRegression()
logreg_model.fit(X_train_c, y_train_c)
y_pred_logreg = logreg_model.predict(X_test_c)
```

In [31]:

```python
# Random Forest Classifier
rf_clf = RandomForestClassifier(n_estimators=100, random_state=42)
rf_clf.fit(X_train_c, y_train_c)
y_pred_rf_clf = rf_clf.predict(X_test_c)
```

In [33]:

```python
# Classification model evaluation
classification_metrics = {
    "Logistic Regression Accuracy": accuracy_score(y_test_c, y_pred_logreg),
    "Random Forest Accuracy": accuracy_score(y_test_c, y_pred_rf_clf)
}
```

In [35]:

```python
# Display Classification Report
# Display classification reports
print("Classification Report for Logistic Regression:")
print(classification_report(y_test_c, y_pred_logreg))
print("\nClassification Report for Random Forest:")
print(classification_report(y_test_c, y_pred_rf_clf))


# Display classification model evaluation metrics
print("\nClassification Model Accuracy:")
print(classification_metrics)
```

```
Classification Report for Logistic Regression:
              precision    recall  f1-score   support

           0       0.94      0.90      0.92       146
           1       0.88      0.93      0.90       122

    accuracy                           0.91       268
   macro avg       0.91      0.91      0.91       268
weighted avg       0.91      0.91      0.91       268


Classification Report for Random Forest:
              precision    recall  f1-score   support

           0       0.90      0.97      0.93       146
           1       0.95      0.87      0.91       122

    accuracy                           0.92       268
   macro avg       0.93      0.92      0.92       268
weighted avg       0.92      0.92      0.92       268


Classification Model Accuracy:
{'Logistic Regression Accuracy': 0.9104477611940298, 'Random Forest Accuracy': 0.9216417910447762}
```

In [176…

```python
print('Logistic regression accuracy is less accurate than random forest accuracy')
```

```
Logistic regression accuracy is less accurate than random forest accuracy
```

In [ ]:

In [ ]:

```python
# Realtime risk assessment
```

In [ ]:

In [37]:

```python
if 'smoker' in df.columns:
    df['smoker'] = df['smoker'].map({'yes': 1, 'no': 0})  # Binary encoding for smoker
if 'sex' in df.columns:
    df['sex'] = df['sex'].map({'male': 1, 'female': 0})  # Binary encoding for gender
if 'region' in df.columns:
    df['region'] = pd.Categorical(df['region']).codes  # Convert region to numerical labels

# Ensure 'charges' column exists before processing high risk classification
if 'charges' in df.columns:
    high_cost_threshold = df['charges'].median()
    df['high_risk'] = (df['charges'] > high_cost_threshold).astype(int)  # Binary classification target
else:
    raise KeyError("Column 'charges' not found in dataset")
```

In [39]:

```python
# Define high-cost risk threshold
high_cost_threshold = df['charges'].median()
df['high_risk'] = (df['charges'] > high_cost_threshold).astype(int)  # Binary classification target
```

In [41]:

```python
# Features and target for classification
X_class = df[['age', 'bmi', 'children', 'smoker']]
y_class = df['high_risk']
```

In [43]:

```python
# Split data
X_train_c, X_test_c, y_train_c, y_test_c = train_test_split(X_class, y_class, test_size=0.2, random_state=42)
```

In [45]:

```python
# Random Forest Classifier
rf_clf = RandomForestClassifier(n_estimators=100, random_state=42)
rf_clf.fit(X_train_c, y_train_c)
```

Out[45]:

```
RandomForestClassifier

RandomForestClassifier(random_state=42)
```

In [47]:

```python
# Save the trained model
joblib.dump(rf_clf, "risk_assessment_model.pkl")
```

Out[47]:

```
['risk_assessment_model.pkl']
```

In [55]:

```python
# Apply predictions to the entire dataset
df['predicted_risk'] = rf_clf.predict(X_class)
df['predicted_risk_label'] = df['predicted_risk'].map({1: 'High Risk', 0: 'Low Risk'})
```

In [67]:

```python
# Save the dataset with predictions for Tableau
df.to_csv("insurance_risk_predictions.csv", index=False)
```

In [83]:

```python
# Real-time risk assessment function
def predict_risk(age, bmi, children, smoker):
    model = joblib.load("risk_assessment_model.pkl")
    input_data = pd.DataFrame([[age, bmi, children, smoker]], columns=['age', 'bmi', 'children', 'smoker'])
    prediction = model.predict(input_data)
    return "High Risk" if prediction[0] == 1 else "Low Risk"
```

In [85]:

```python
# Example Usage
print(predict_risk(45, 30.5, 2, 1))  # Predicts risk based on input values
```

```
Low Risk
```

In [87]:

```python
# Display count of high risk and low risk individuals
high_risk_count = (df['predicted_risk'] == 1).sum()
low_risk_count = (df['predicted_risk'] == 0).sum()
```

```python
print(f"High Risk Count: {high_risk_count}")
print(f"Low Risk Count: {low_risk_count}")
```

```
High Risk Count: 647
Low Risk Count: 691
```