kqkrpr8wf

February 15, 2025

0.0.1 Machine Learning Workflow for Auto Insurance Dataset

My dataset includes features related to **customer demographics, prior insurance history, claims data, premium adjustments, and conversion status**. Depending on my goal, different **ML techniques** will be applied.

0.1 1 Problem Definition

- I want to **predict whether a customer will convert** (buy insurance), it's a **classification problem**.
  - **Target Variable:** `Conversion_Status` (Binary: 0 = No, 1 = Yes)

  - **ML Techniques:** Logistic Regression, Random Forest, XGBoost, Neural Networks
- I want to **predict premium adjustment** based on customer history, it's a **regression problem**.
  - **Target Variable:** `Premium_Adjustment_Credit` or `Premium_Adjustment_Region` (Continuous Values)

  - **ML Techniques:** Linear Regression, Random Forest Regressor, XGBoost Regressor

---

0.2 2 Exploratory Data Analysis (EDA)

Goal: Understand dataset characteristics and relationships between variables.
Techniques:
- **Check for Missing Values:** `df.isnull().sum()`
- **Feature Distributions:** Histograms, Boxplots
- **Correlation Analysis:** `df.corr()` + Heatmap
- **Categorical Data Visualization:** Countplots for `Marital_Status`, `Policy_Type`, etc.
- **Outlier Detection:** IQR method

0.3 3 Data Preprocessing & Feature Engineering

Goal: Prepare data for ML models by encoding categorical variables, handling missing values, and scaling numerical data.

Techniques:
- **Encoding Categorical Variables:**
- One-Hot Encoding (`Marital_Status`, `Region`, `Policy_Type`)
- Label Encoding (`Claims_Severity`: Low → 0, Medium → 1, High → 2)
- **Handling Missing Values:**
- Impute `NaN` values with **mean/median** (numerical) or **mode** (categorical).
- **Feature Scaling:**
- **Standardization (`StandardScaler`)** for models like Logistic Regression, SVM
- **Normalization (`MinMaxScaler`)** for Neural Networks
- **Feature Selection:**
- Remove highly correlated features (e.g., `Premium_Adjustment_Credit` and `Premium_Adjustment_Region` may be related)
- Use **Recursive Feature Elimination (RFE)** to find important features

---

0.4 4 Model Selection & Training

Based on the problem type:

0.4.1 Classification (Predicting `Conversion_Status`)

- **Logistic Regression** (Baseline Model)
- **Random Forest Classifier** (Handles categorical & numerical features well)
- **XGBoost/LightGBM** (Best for structured insurance data)
- **Artificial Neural Networks (ANN)** (For deep learning-based approaches)

Pipeline:
1. Train-Test Split: `train_test_split(df, test_size=0.2, stratify=df['Conversion_Status'])`
2. Model Training: `RandomForestClassifier(n_estimators=100)`
3. Evaluate with **Accuracy, Precision, Recall, F1-score, ROC-AUC**

0.4.2 Regression (Predicting `Premium_Adjustment_Credit`)

- **Linear Regression** (Baseline)
- **Random Forest Regressor** (Handles nonlinear relationships)
- **XGBoost Regressor** (For boosting performance)

Pipeline:
1. Train-Test Split: `train_test_split(df, test_size=0.2, random_state=42)`
2. Model Training: `XGBRegressor(n_estimators=500, learning_rate=0.1)`
3. Evaluate using **RMSE, R^2 Score, MAE**

0.5 5 Model Evaluation & Optimization

Classification Metrics:
- `classification_report(y_test, y_pred)` (Accuracy, Precision, Recall, F1-score)
- `roc_auc_score(y_test, y_pred_prob)` (For imbalanced data)

Regression Metrics:
- `mean_squared_error(y_test, y_pred, squared=False)` (RMSE)
- `r2_score(y_test, y_pred)` (Explained Variance)

Hyperparameter Tuning:
- `GridSearchCV` (For Logistic Regression, Random Forest)
- `RandomizedSearchCV` (For XGBoost, LightGBM)

0.6 6 Model Deployment (Optional)

If you want to **deploy the model**, I can:
- **Save the model:** `pickle.dump(model, open('model.pkl', 'wb'))`
- **Deploy with Flask/FastAPI** for API-based predictions
- **Build a Dashboard with Streamlit/Dash**

0.6.1 Conclusion

My dataset allows for **multiple ML applications**, from customer conversion prediction to premium optimization. **Tree-based models (Random Forest, XGBoost) will likely perform best** due to structured data.

[16]:
```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
import pandas as pd
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report, accuracy_score,␣
↪confusion_matrix, roc_auc_score
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
```

```python
from catboost import CatBoostClassifier
```

[ ]:

[ ]:

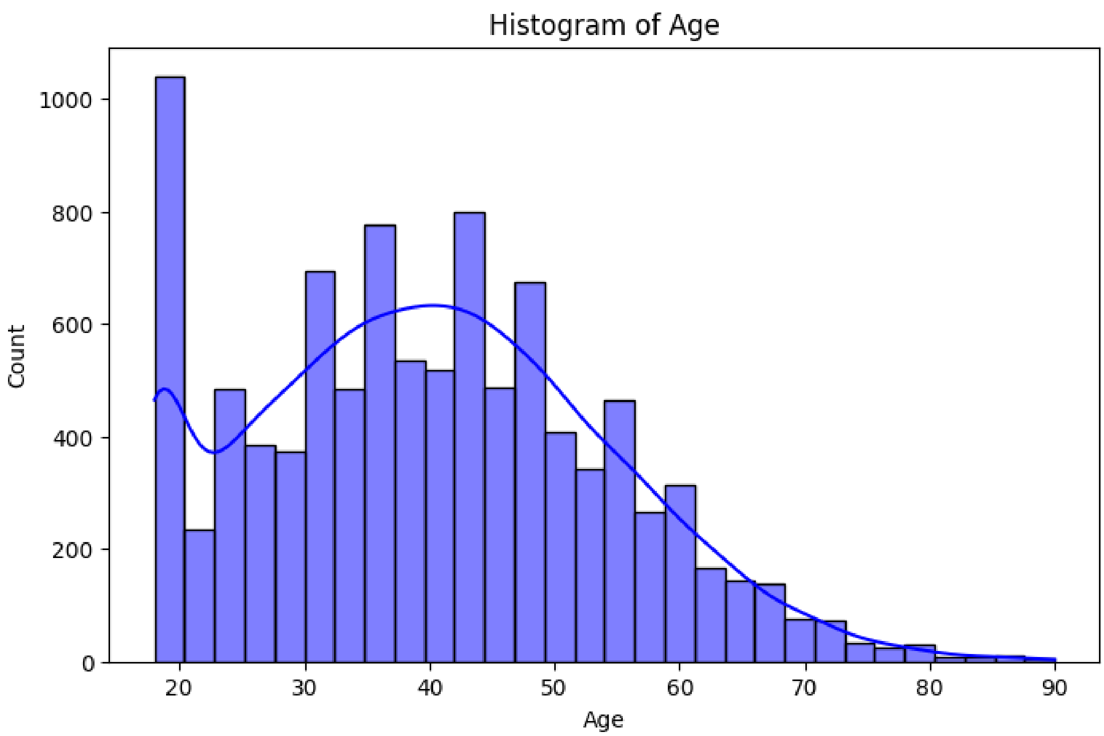
[3]:
```python
df = pd.read_csv("/kaggle/input/insurance-data-personal-auto-line-of-business/
↪synthetic_insurance_data.csv")
```

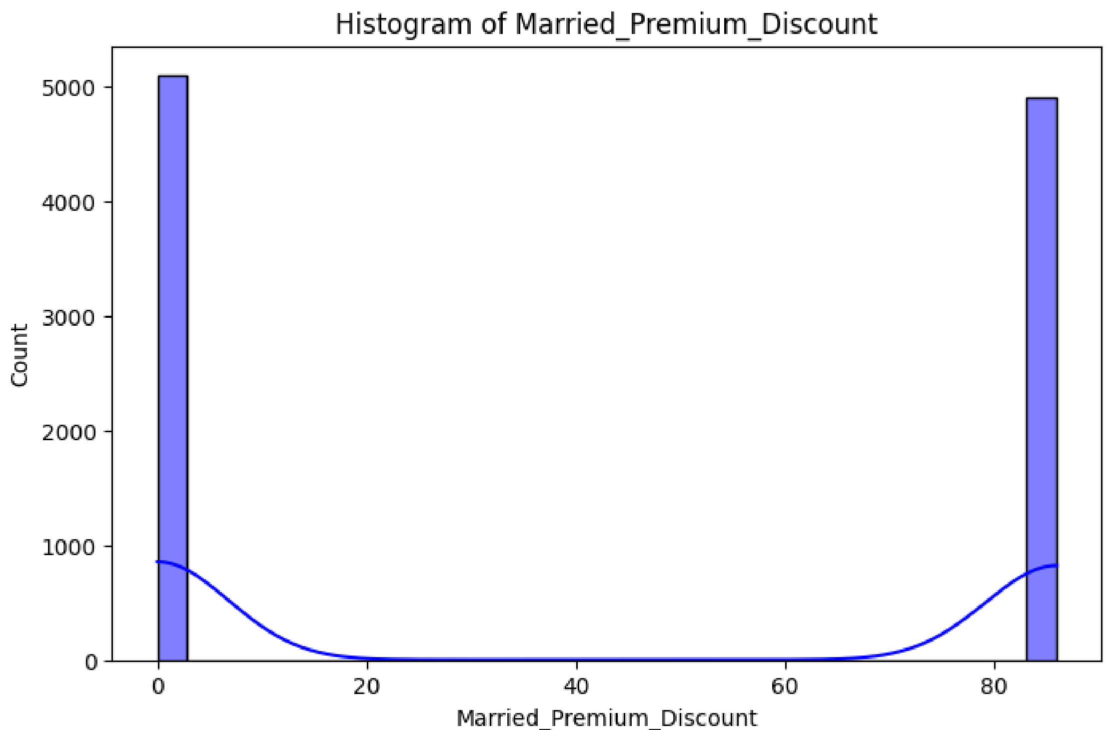
[4]:
```python
df.head()
```

[4]:
```
   Age  Is_Senior Marital_Status  Married_Premium_Discount Prior_Insurance  \
0   47          0        Married                        86       1-5 years
1   37          0        Married                        86       1-5 years
2   49          0        Married                        86       1-5 years
3   62          1        Married                        86        >5 years
4   36          0         Single                         0        >5 years

   Prior_Insurance_Premium_Adjustment  Claims_Frequency Claims_Severity  \
0                                  50                 0             Low
1                                  50                 0             Low
2                                  50                 1             Low
3                                   0                 1             Low
4                                   0                 2             Low

   Claims_Adjustment    Policy_Type  …  Time_Since_First_Contact  \
0                  0  Full Coverage  …                        10
1                  0  Full Coverage  …                        22
2                 50  Full Coverage  …                        28
3                 50  Full Coverage  …                         4
4                100  Full Coverage  …                        14

   Conversion_Status  Website_Visits  Inquiries  Quotes_Requested  \
0                  0               5          1                 2
1                  0               5          1                 2
2                  0               4          4                 1
3                  1               6          2                 2
4                  1               8          4                 2

   Time_to_Conversion Credit_Score  Premium_Adjustment_Credit    Region  \
0                  99          704                        -50  Suburban
1                  99          726                        -50     Urban
2                  99          772                        -50     Urban
3                   2          809                        -50     Urban
4                  10          662                         50  Suburban

   Premium_Adjustment_Region
```

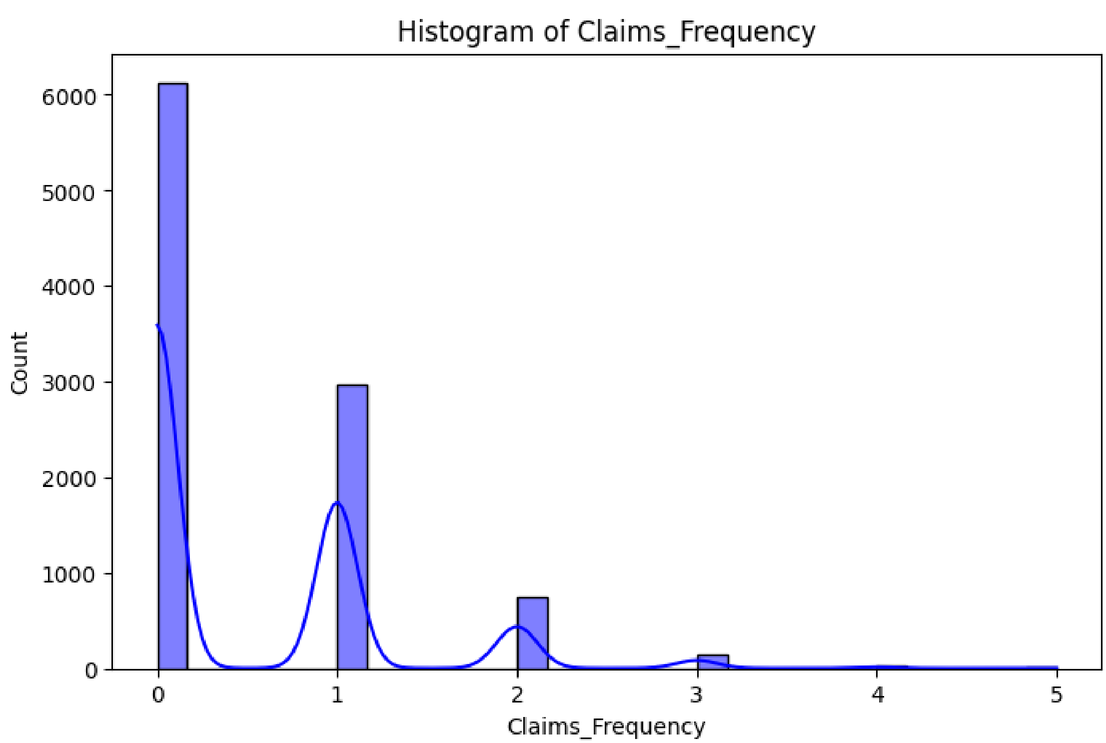
```
0                             50
1                            100
2                            100
3                            100
4                             50

[5 rows x 27 columns]
```

[5]:
```python
# Histogram
num_cols = ['Age', 'Married_Premium_Discount', 'Claims_Frequency',␣
 ↪'Credit_Score']
for col in num_cols:
    plt.figure(figsize=(8, 5))
    sns.histplot(df[col], bins=30, kde=True, color='blue')
    plt.title(f'Histogram of {col}')
    plt.show()
```

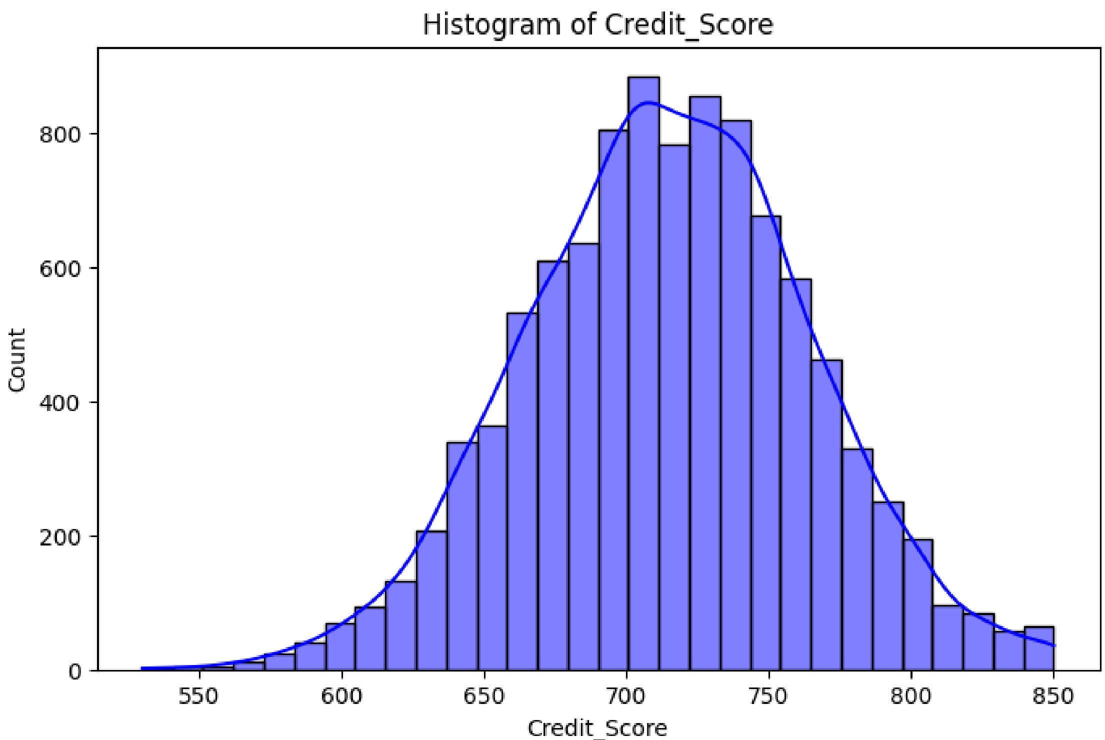




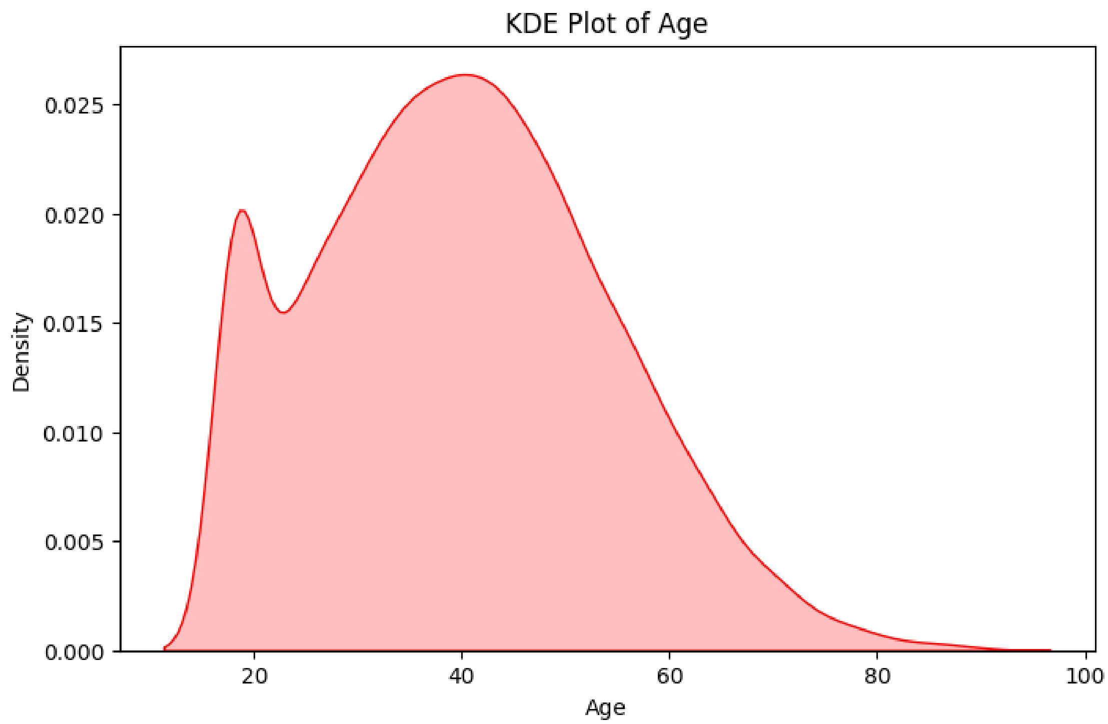
Histogram of Married_Premium_Discount
Count
Married_Premium_Discount
0
1000
2000
3000
4000
5000
0
20
40
60
80



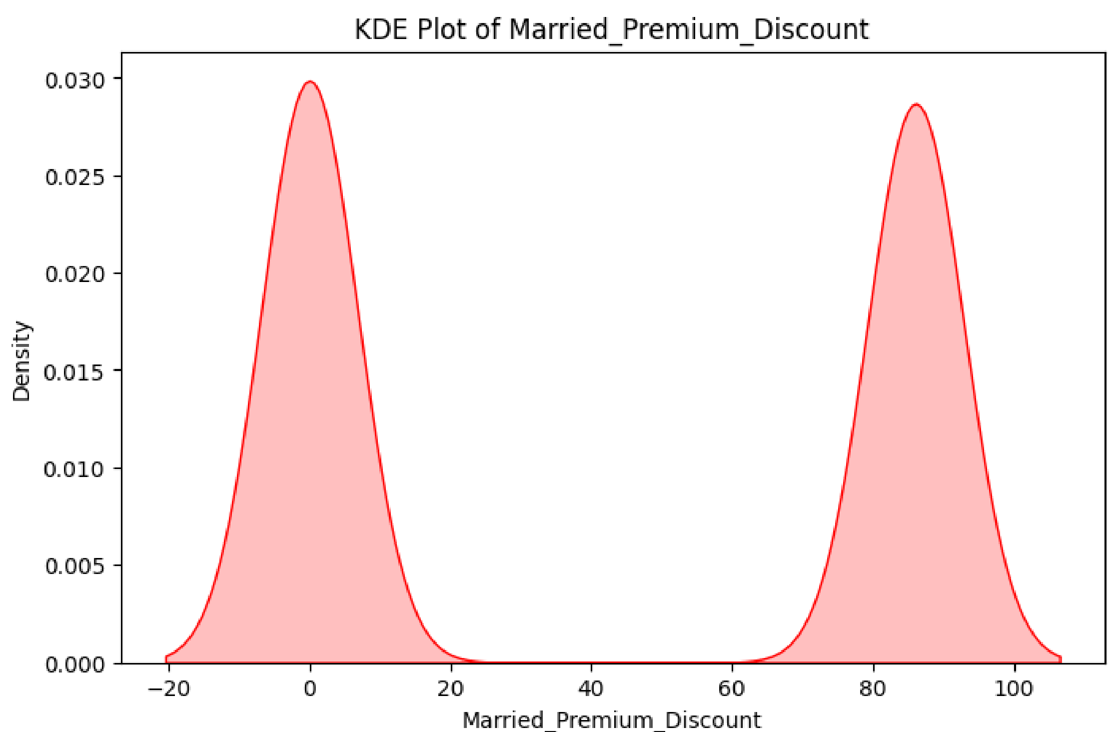
Histogram of Claims_Frequency
Count
Claims_Frequency
0
1000
2000
3000
4000
5000
6000
0
1
2
3
4
5



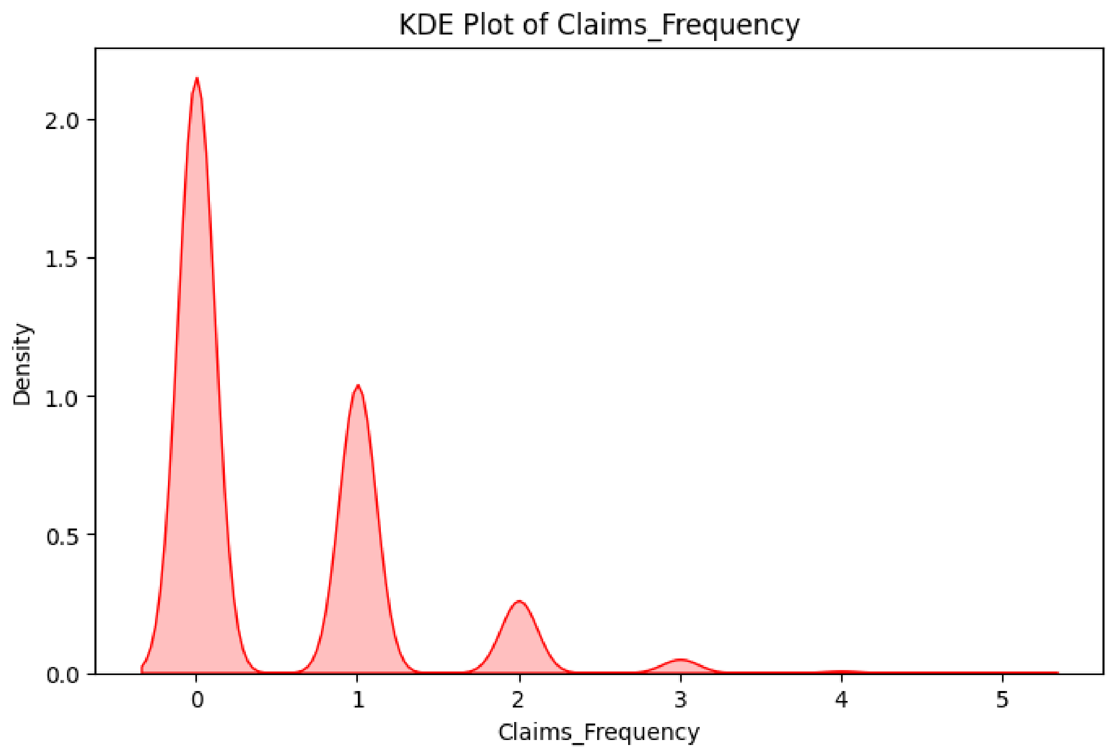
[6]:
```
# KDE Plots
for col in num_cols:
    plt.figure(figsize=(8, 5))
    sns.kdeplot(df[col], fill=True, color='red')
    plt.title(f'KDE Plot of {col}')
    plt.show()
```

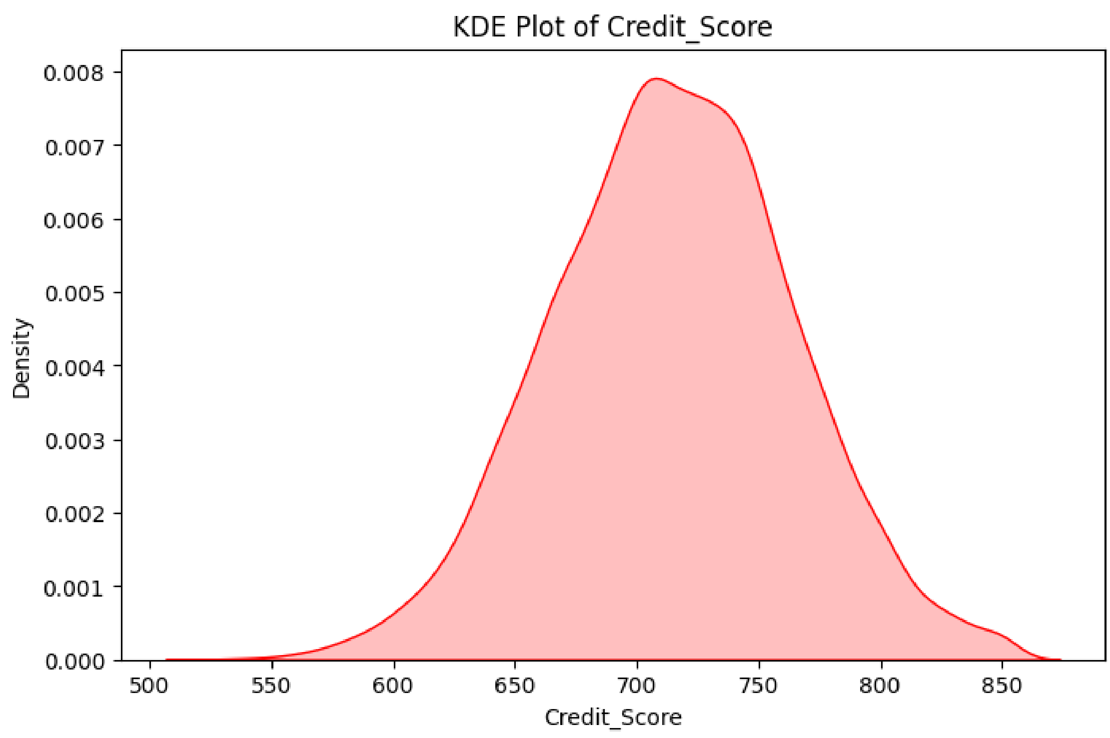


KDE Plot of Age
Density
0.000
0.005
0.010
0.015
0.020
0.025
20
40
60
80
100
Age



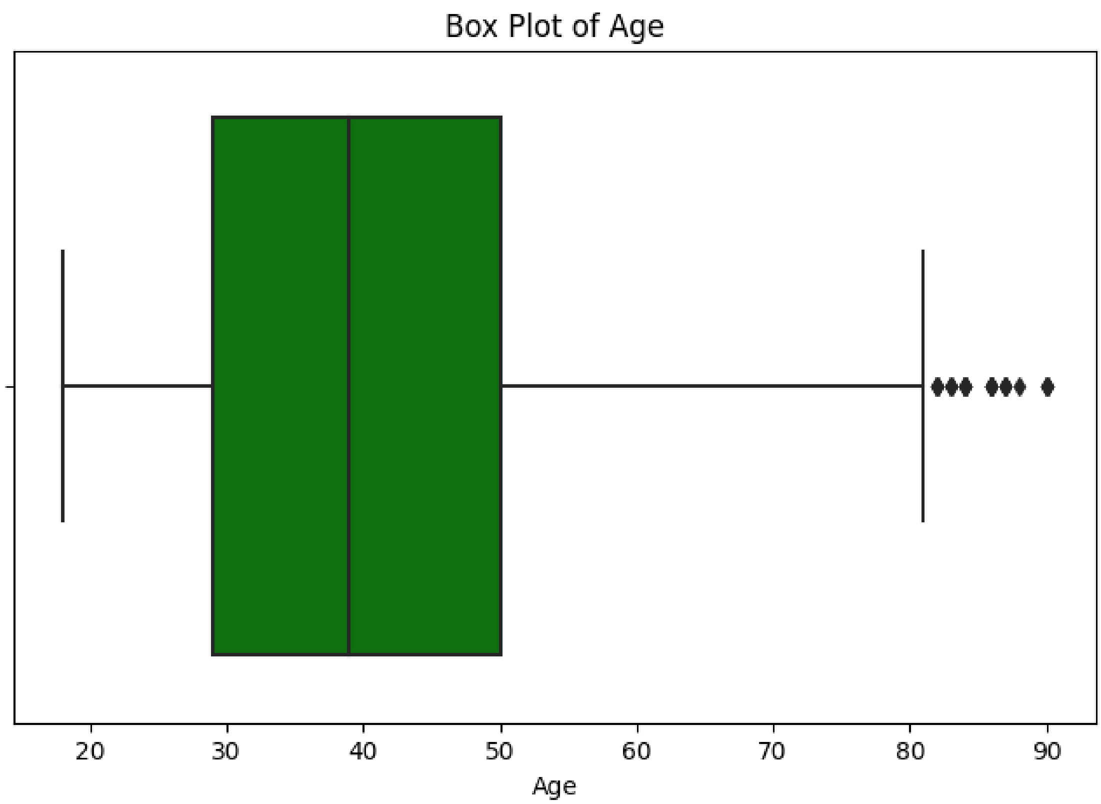
KDE Plot of Married_Premium_Discount
Density
0.000
0.005
0.010
0.015
0.020
0.025
0.030
−20
0
20
40
60
80
100
Married_Premium_Discount

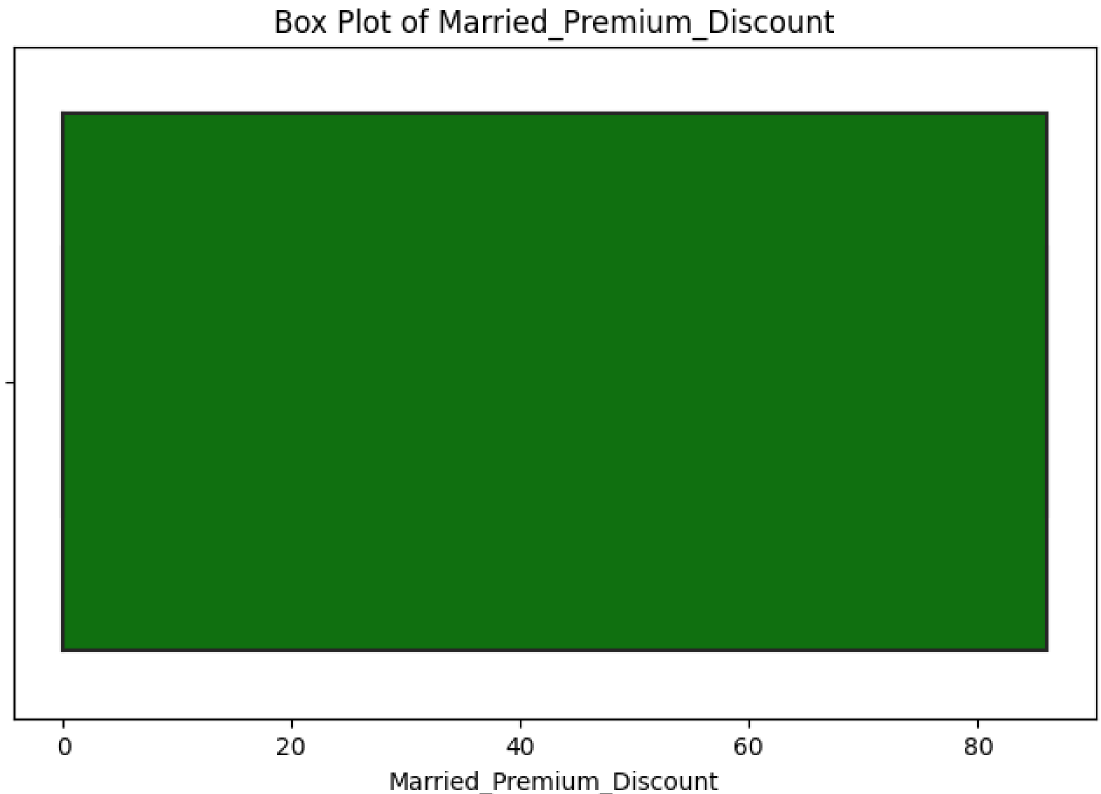

KDE Plot of Claims_Frequency
Density
2.0
1.5
1.0
0.5
0.0
0
1
2
3
4
5
Claims_Frequency

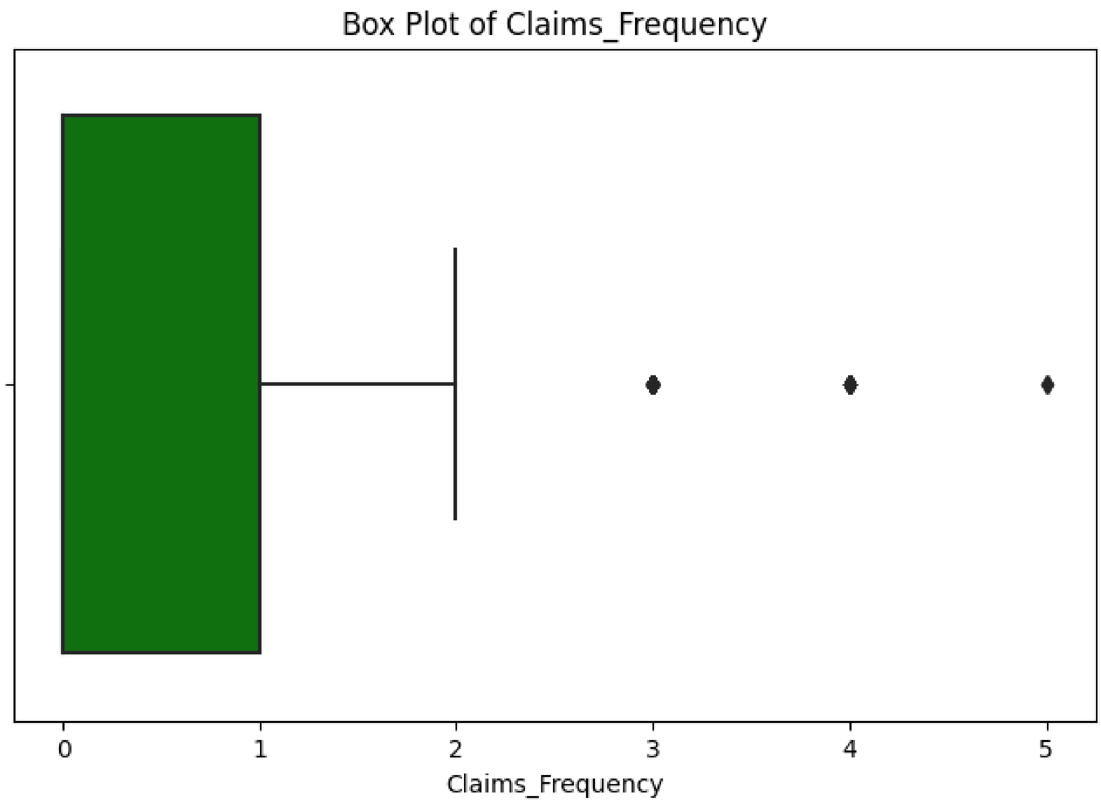

KDE Plot of Credit_Score
Density
0.008
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0.000
500
550
600
650
700
750
800
850
Credit_Score

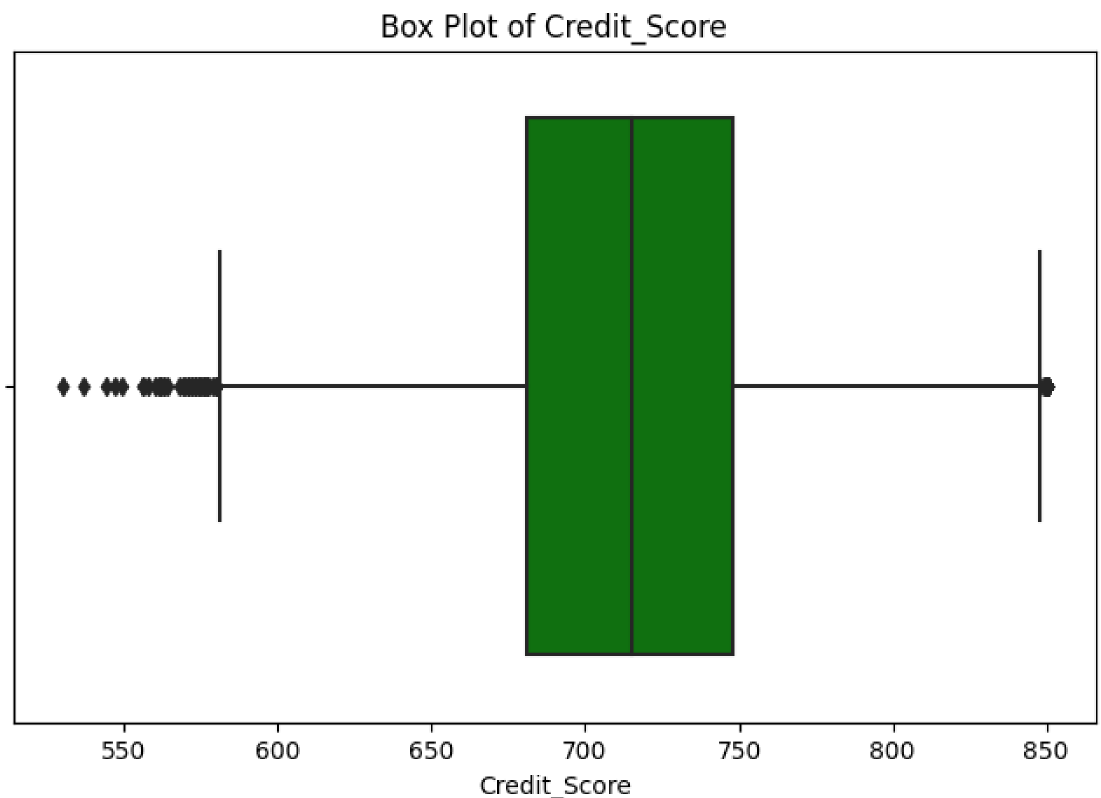
[7]:

```
# Box Plots
for col in num_cols:
    plt.figure(figsize=(8, 5))
    sns.boxplot(x=df[col], color='green')
    plt.title(f'Box Plot of {col}')
    plt.show()
```




Box Plot of Married_Premium_Discount
0
20
40
60
80
Married_Premium_Discount

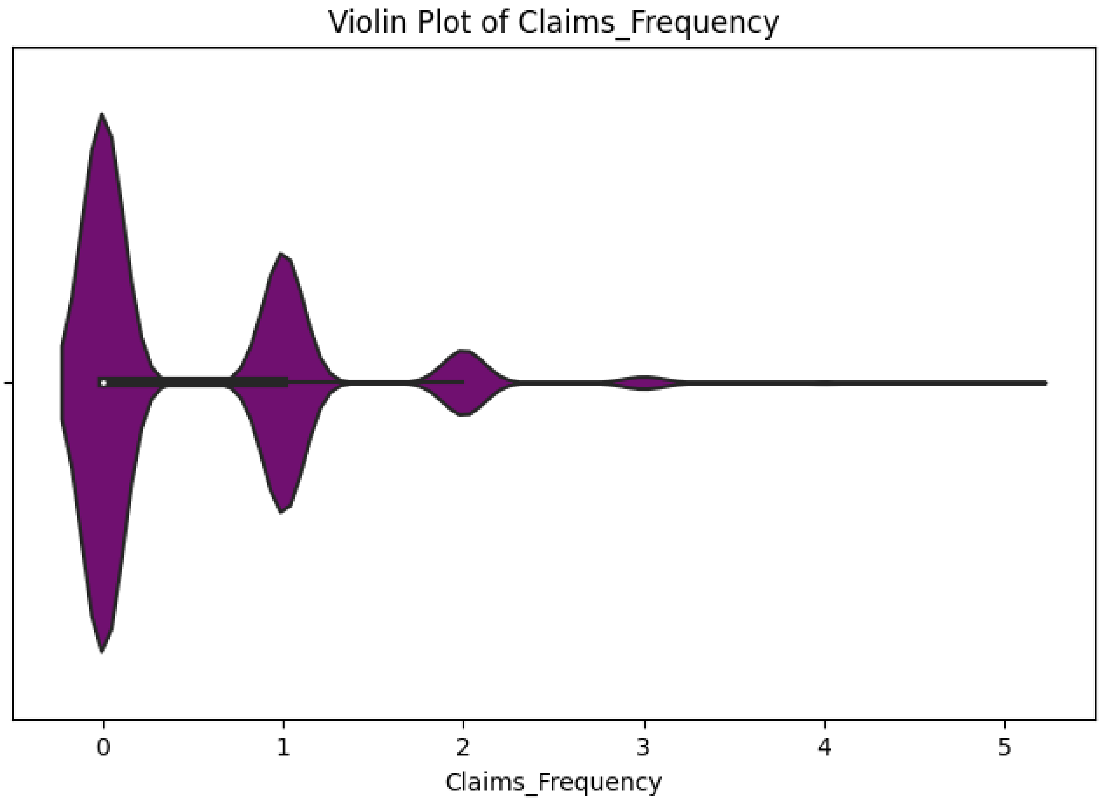

Box Plot of Claims_Frequency
0
1
2
3
4
5
Claims_Frequency

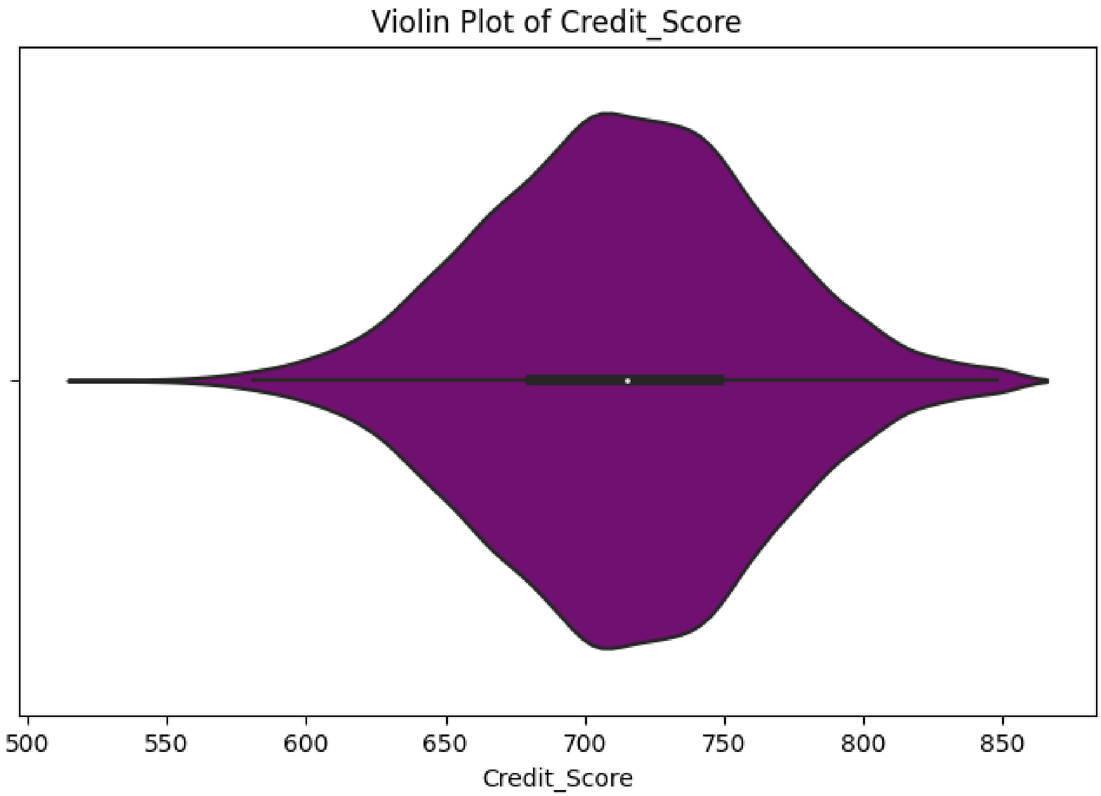


[8]:
```
# Violin Plots
for col in num_cols:
    plt.figure(figsize=(8, 5))
    sns.violinplot(x=df[col], color='purple')
    plt.title(f'Violin Plot of {col}')
    plt.show()
```

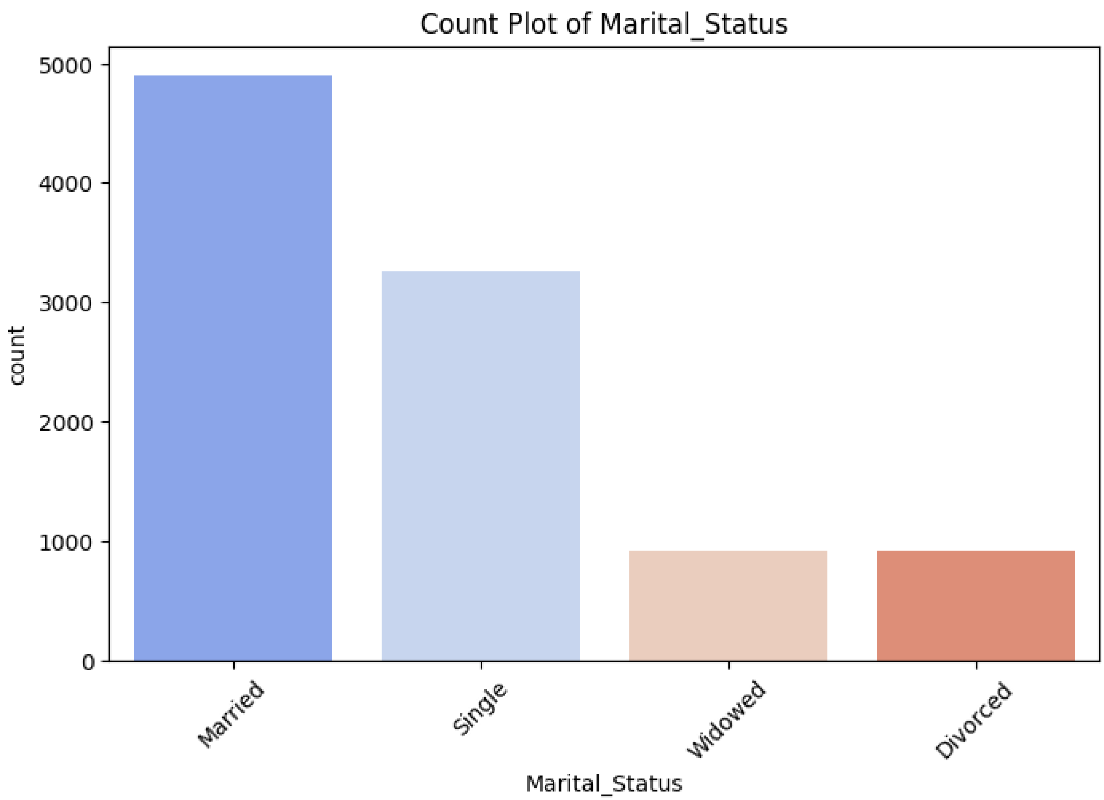

Violin Plot of Age
10
20
30
40
50
60
70
80
90
Age

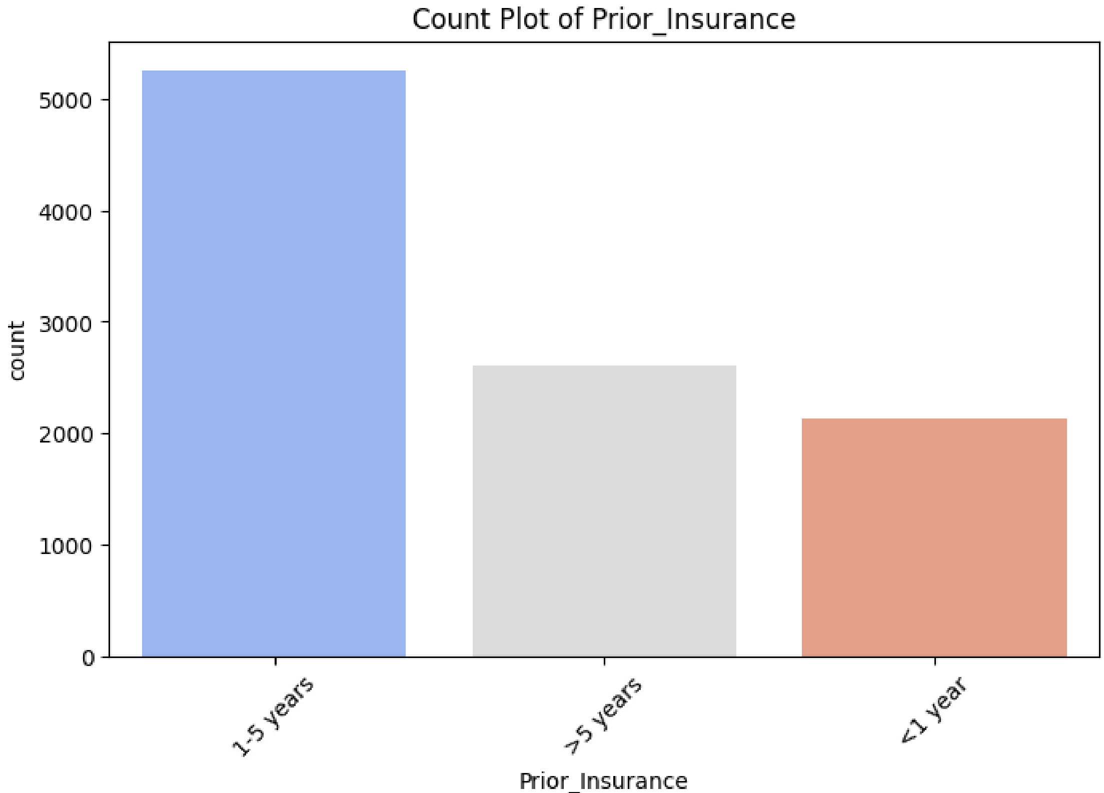

Violin Plot of Married_Premium_Discount
0
20
40
60
80
100
Married_Premium_Discount

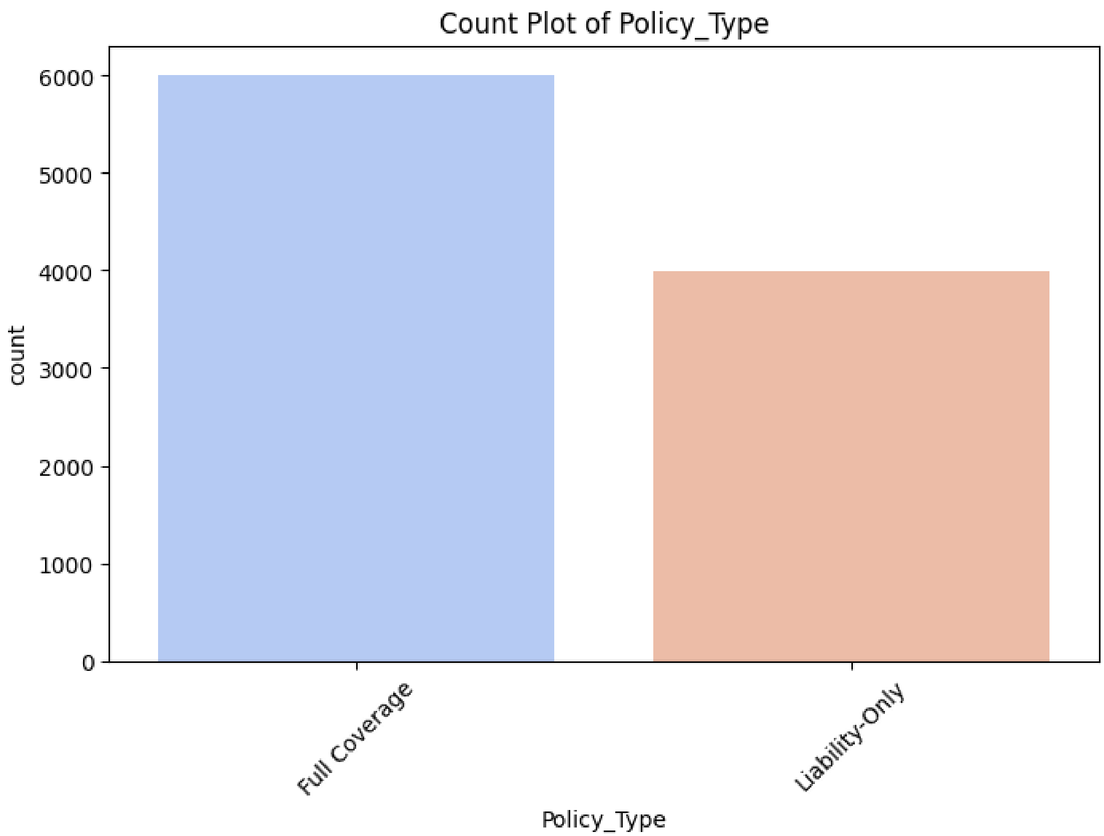

Violin Plot of Claims_Frequency
0
1
2
3
4
5
Claims_Frequency

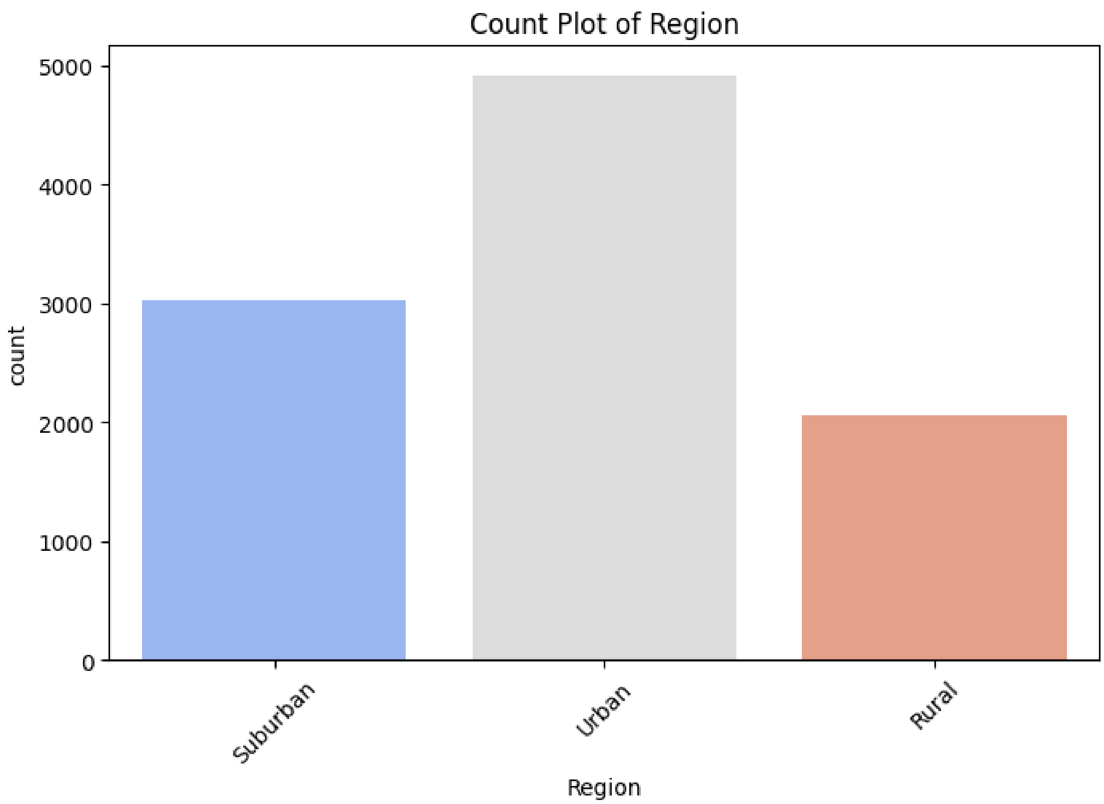


[9]:
```
# Count Plots (Categorical Data)
categorical_cols = ['Marital_Status', 'Prior_Insurance', 'Policy_Type',␣
↪'Region']
for col in categorical_cols:
    plt.figure(figsize=(8, 5))
    sns.countplot(x=df[col], palette='coolwarm')
    plt.title(f'Count Plot of {col}')
    plt.xticks(rotation=45)
    plt.show()
```

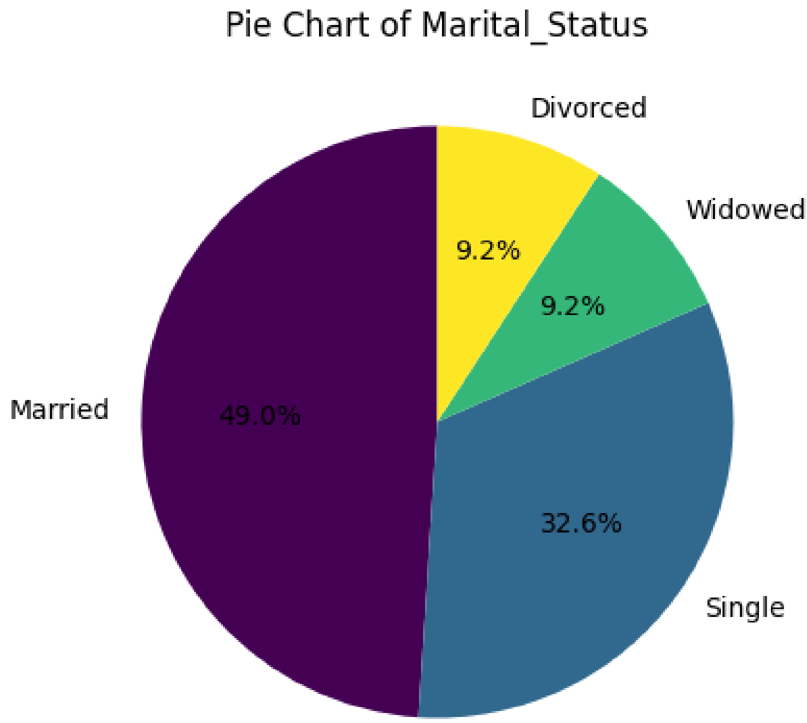

Count Plot of Marital_Status
count
5000
4000
3000
2000
1000
0
Married
Single
Widowed
Divorced
Marital_Status


Count Plot of Prior_Insurance
count
5000
4000
3000
2000
1000
0
1-5 years
>5 years
<1 year
Prior_Insurance

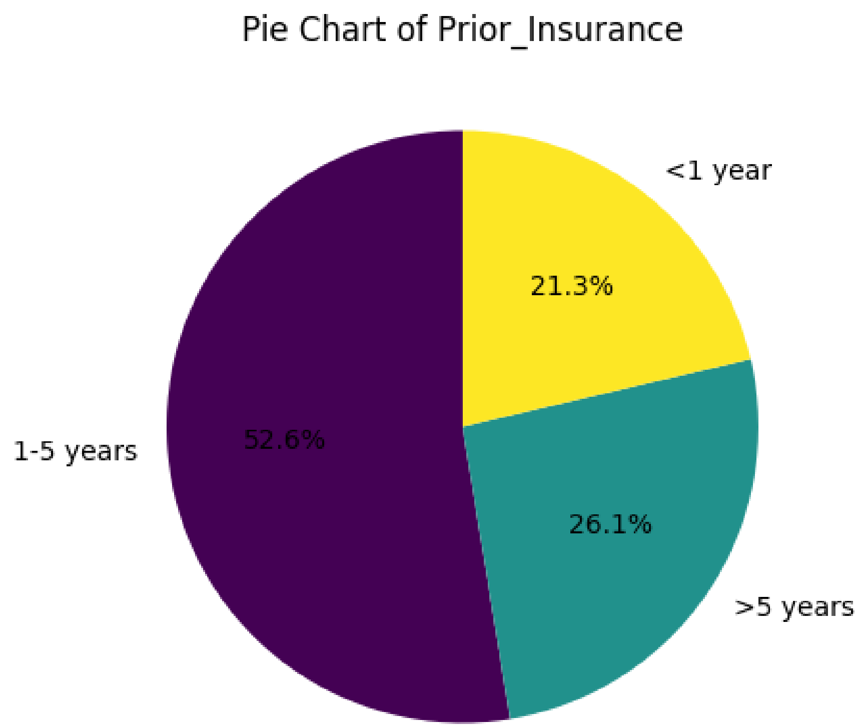

Count Plot of Policy_Type
6000
5000
4000
3000
2000
1000
0
count
Full Coverage
Liability-Only
Policy_Type

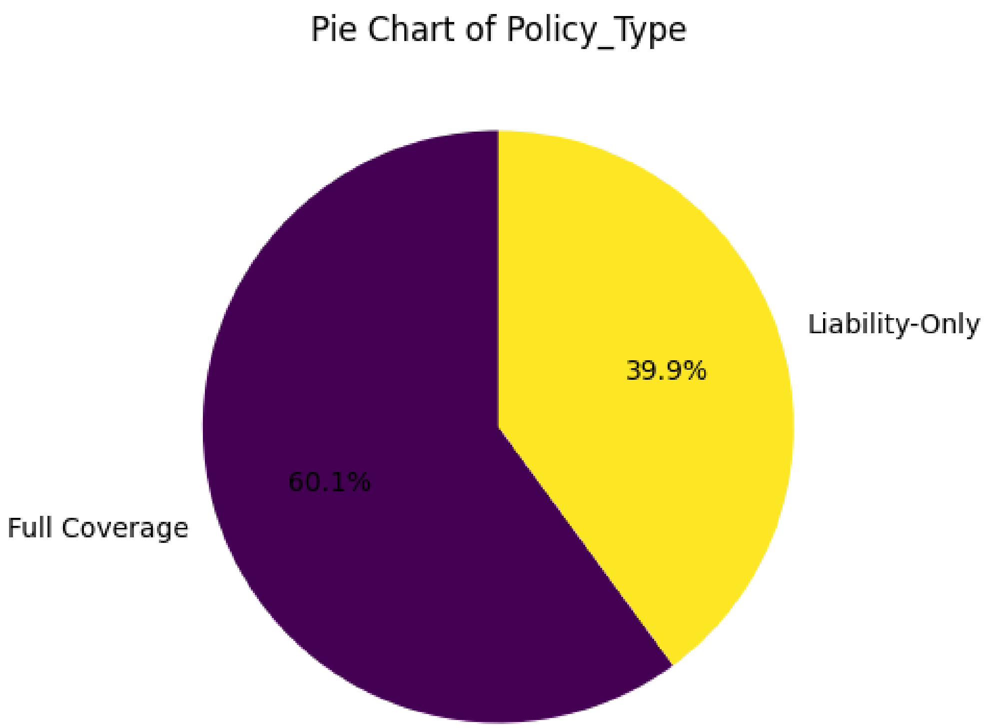


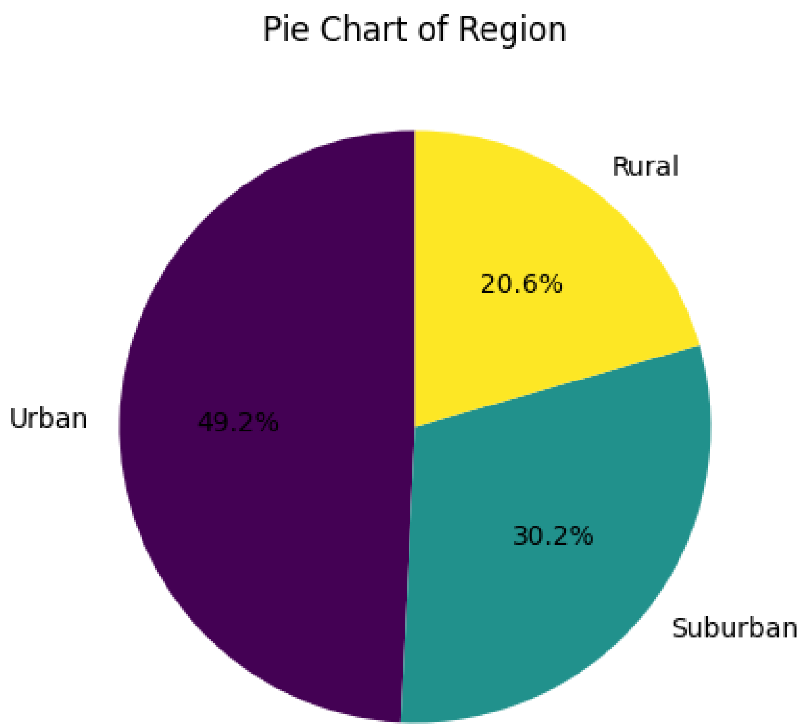
[10]:
```
# Pie Charts
for col in categorical_cols:
    plt.figure(figsize=(8, 5))
    df[col].value_counts().plot.pie(autopct='%1.1f%%', startangle=90,␣
 ↪cmap='viridis')
    plt.title(f'Pie Chart of {col}')
    plt.ylabel('')
    plt.show()
```



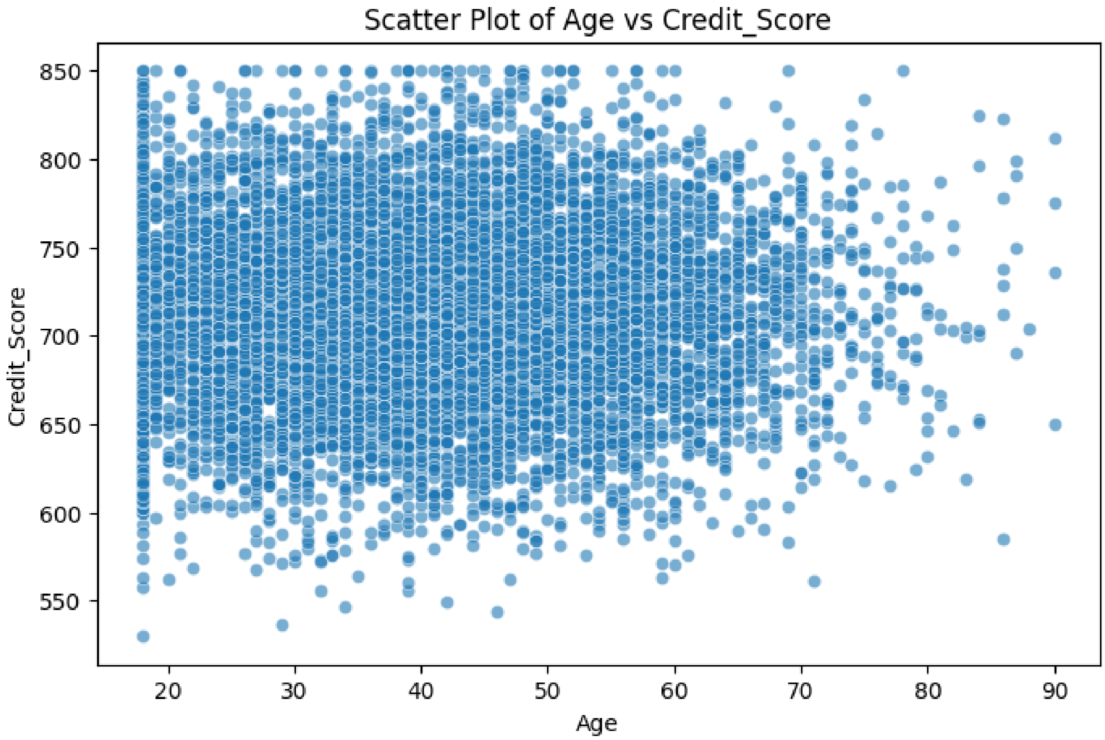
Pie Chart of Marital_Status
Divorced
Widowed
Married
Single
9.2%
9.2%
49.0%
32.6%

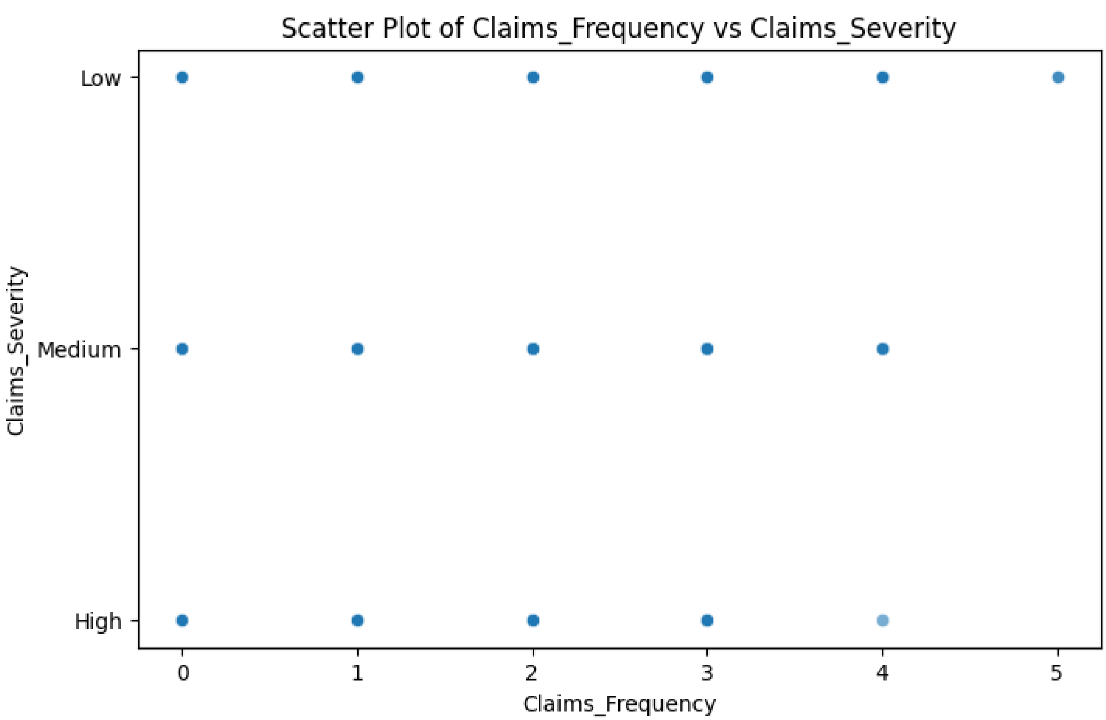

Pie Chart of Prior_Insurance
<1 year
21.3%
1-5 years
52.6%
26.1%
>5 years


Pie Chart of Policy_Type
Liability-Only
39.9%
60.1%
Full Coverage

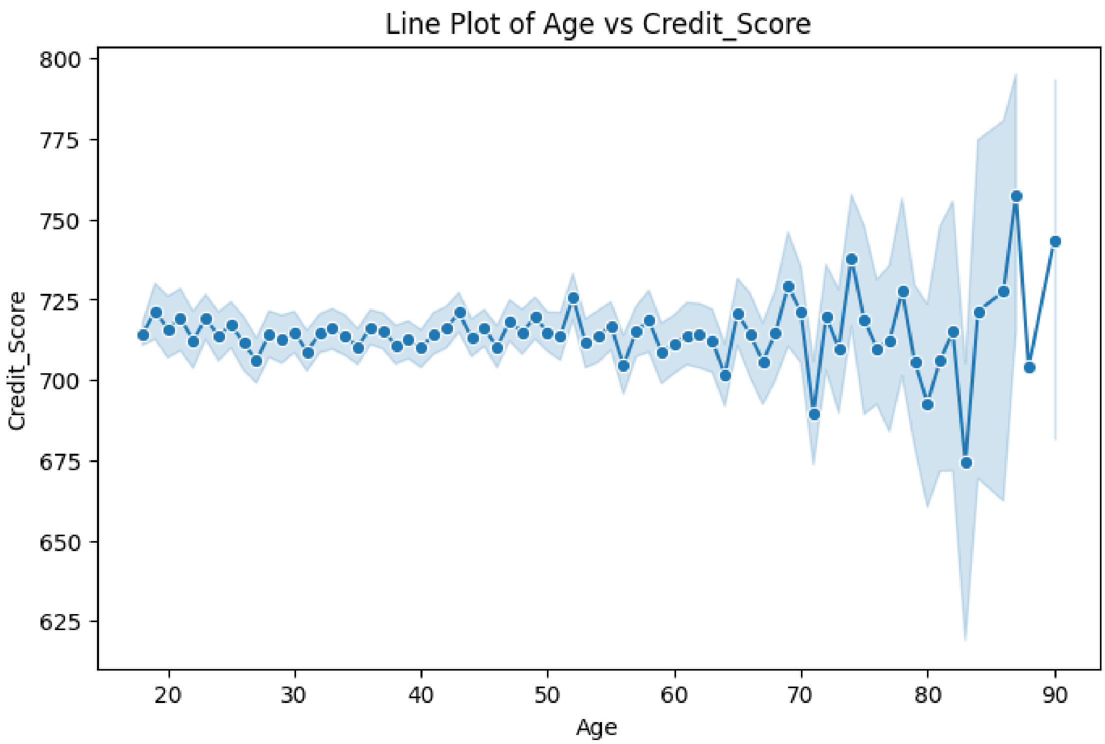


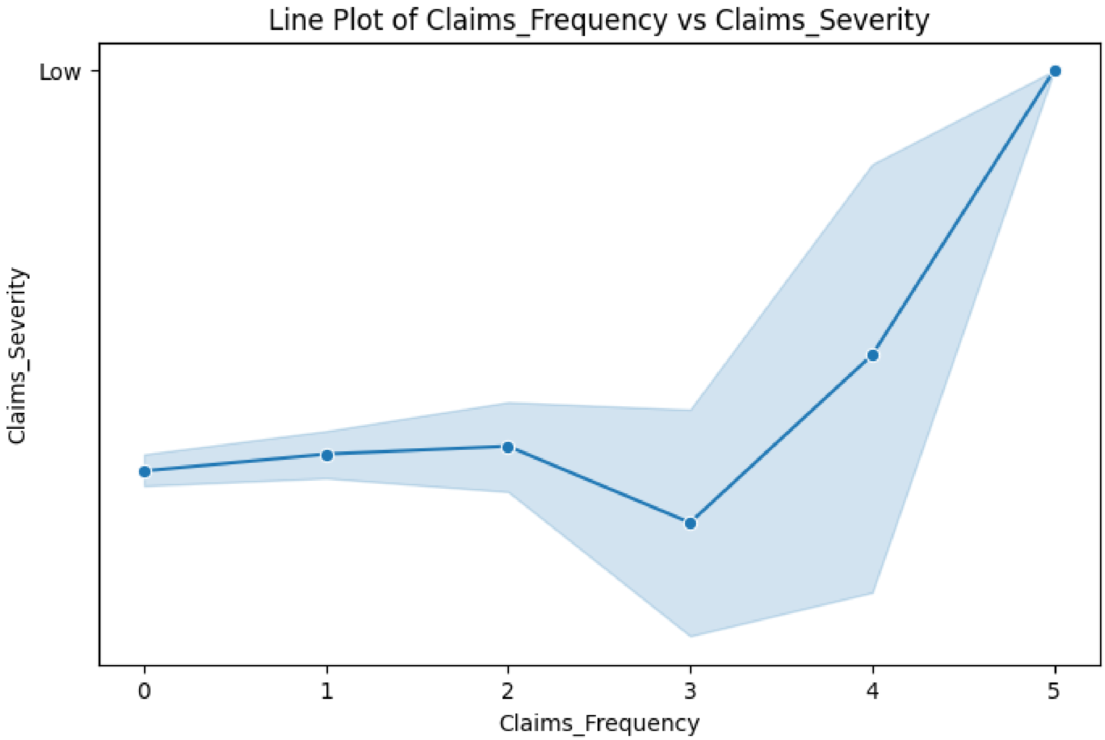
[11]:
```
# Scatter Plots
bivariate_cols = [('Age', 'Credit_Score'), ('Claims_Frequency',␣
↪'Claims_Severity')]
for x, y in bivariate_cols:
    plt.figure(figsize=(8, 5))
    sns.scatterplot(x=df[x], y=df[y], alpha=0.6)
    plt.title(f'Scatter Plot of {x} vs {y}')
    plt.show()
```


Scatter Plot of Age vs Credit_Score
850
800
750
700
650
600
550
Credit_Score
20
30
40
50
60
70
80
90
Age

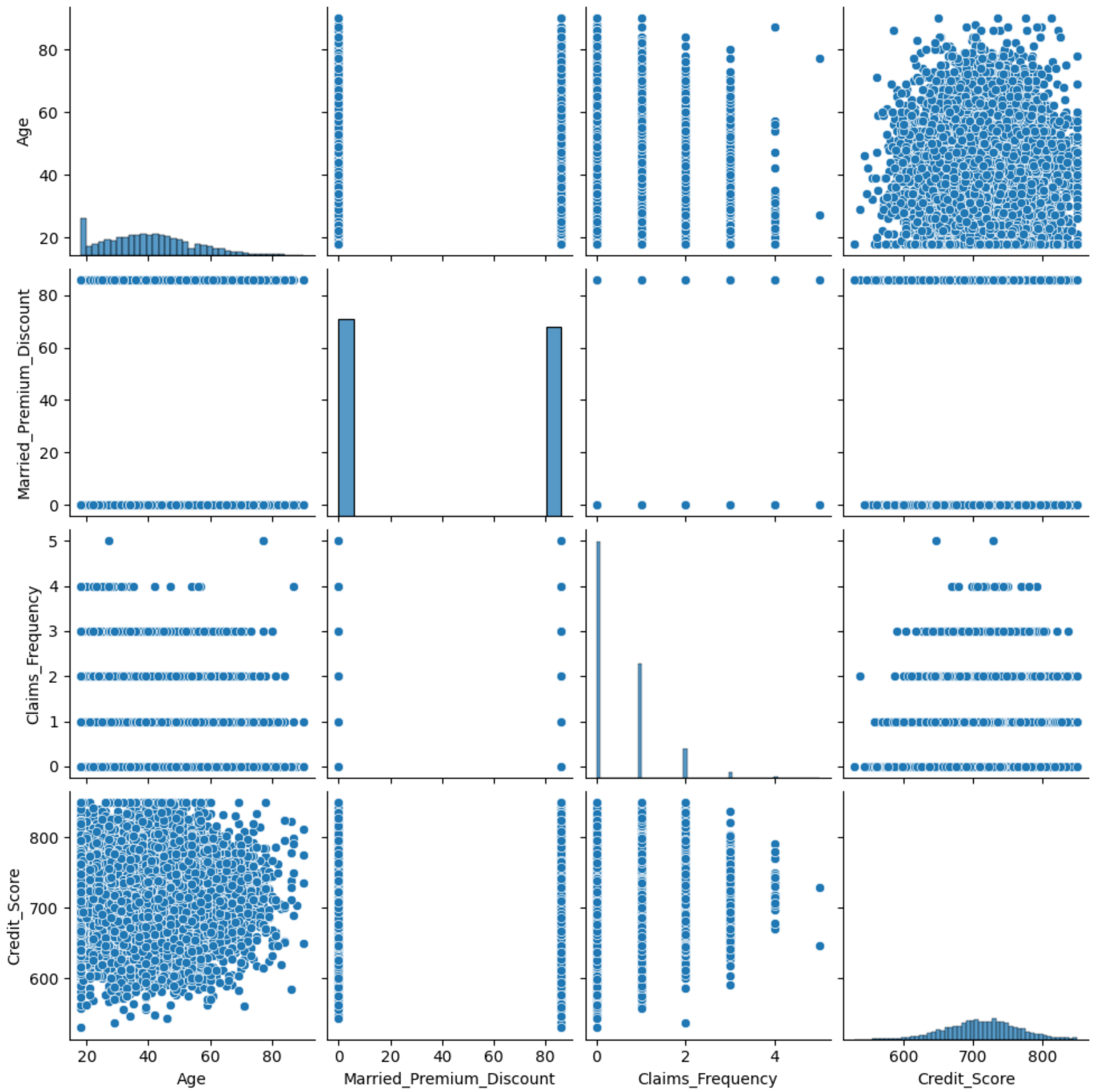

Scatter Plot of Claims_Frequency vs Claims_Severity
Low
Medium
High
Claims_Severity
0
1
2
3
4
5
Claims_Frequency

[12]:
```
# Line Plots
for x, y in bivariate_cols:
    plt.figure(figsize=(8, 5))
    sns.lineplot(x=df[x], y=df[y], marker='o')
    plt.title(f'Line Plot of {x} vs {y}')
    plt.show()
```





[13]:
```
# Pair Plot
sns.pairplot(df[num_cols])
plt.show()
```



[14]:

```
# -------------------------- Outlier Detection & Removal␣
↪-------------------------- #

def detect_outliers(df, cols):
    outliers = {}
    for col in cols:
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        outliers[col] = df[(df[col] < lower_bound) | (df[col] >␣
↪upper_bound)][col]
```

```
    return outliers

# Numerical columns to check for outliers
num_cols = ['Age', 'Married_Premium_Discount', 'Claims_Frequency',␣
 ↪'Credit_Score']
outliers = detect_outliers(df, num_cols)
print("Detected Outliers:", outliers)

# Remove outliers
def remove_outliers(df, cols):
    for col in cols:
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        df = df[(df[col] >= lower_bound) & (df[col] <= upper_bound)]
    return df

df = remove_outliers(df, num_cols)
print("Outliers removed. New dataset shape:", df.shape)
```

```
Detected Outliers: {'Age': 209     90
478     86
1615    87
1957    87
2305    86
2506    84
2521    84
2895    90
3241    83
3716    88
3982    87
4047    82
4870    86
4997    86
5157    83
5224    84
5673    82
5796    86
5846    83
6738    84
6891    90
7223    86
7330    84
7872    82
7874    82
```

```
8165    84
8248    90
8726    87
Name: Age, dtype: int64, 'Married_Premium_Discount': Series([], Name:
Married_Premium_Discount, dtype: int64), 'Claims_Frequency': 68      3
260     4
284     3
392     4
452     3
       ..
9868    3
9879    3
9887    3
9890    3
9965    4
Name: Claims_Frequency, Length: 164, dtype: int64, 'Credit_Score': 42      850
94      850
239     850
328     577
384     574
       …
9498    850
9728    576
9805    562
9871    577
9937    850
Name: Credit_Score, Length: 81, dtype: int64}
Outliers removed. New dataset shape: (9728, 27)
```

[25]:

```
# Data preprocessing
le = LabelEncoder()
categorical_cols = ['Marital_Status', 'Prior_Insurance', 'Claims_Severity',␣
↪'Policy_Type', 'Region','Source_of_Lead']
for col in categorical_cols:
    df[col] = le.fit_transform(df[col])

X = df.drop(columns=['Conversion_Status'])  # Target variable
y = df['Conversion_Status']
```

[24]:

```
df['Source_of_Lead']
```

[24]:

```
0         Agent
1        Online
2        Online
3        Online
4         Agent
          …
```

```
9995    Online
9996     Agent
9997     Agent
9998     Agent
9999     Agent
Name: Source_of_Lead, Length: 9728, dtype: object
```

[27]:
```python
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Standardization
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

[28]:
```python
# Define models
models = {
    'Logistic Regression': LogisticRegression(),
    'KNN': KNeighborsClassifier(),
    'Decision Tree': DecisionTreeClassifier(),
    'Random Forest': RandomForestClassifier(),
    'SVM': SVC(probability=True),
    'Naive Bayes': GaussianNB(),
    'Gradient Boosting': GradientBoostingClassifier(),
    'XGBoost': XGBClassifier(),
    'LightGBM': LGBMClassifier(),
    'CatBoost': CatBoostClassifier(verbose=0)
}
```

[29]:
```python
# Train and evaluate models
results = {}
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    auc = roc_auc_score(y_test, model.predict_proba(X_test)[:, 1])
    results[name] = {'Accuracy': acc, 'AUC': auc}
    print(f"{name} Classification Report:\n", classification_report(y_test,␣
 ↪y_pred))
    print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

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

           0       1.00      1.00      1.00       817
           1       1.00      1.00      1.00      1129
```

```
    accuracy                           1.00      1946
   macro avg       1.00      1.00      1.00      1946
weighted avg       1.00      1.00      1.00      1946

Confusion Matrix:
 [[ 817    0]
 [   0 1129]]
KNN Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.93      0.96       817
           1       0.95      0.99      0.97      1129

    accuracy                           0.96      1946
   macro avg       0.97      0.96      0.96      1946
weighted avg       0.96      0.96      0.96      1946

Confusion Matrix:
 [[ 762   55]
 [  15 1114]]
Decision Tree Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       817
           1       1.00      1.00      1.00      1129

    accuracy                           1.00      1946
   macro avg       1.00      1.00      1.00      1946
weighted avg       1.00      1.00      1.00      1946

Confusion Matrix:
 [[ 817    0]
 [   0 1129]]
Random Forest Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       817
           1       1.00      1.00      1.00      1129

    accuracy                           1.00      1946
   macro avg       1.00      1.00      1.00      1946
weighted avg       1.00      1.00      1.00      1946

Confusion Matrix:
 [[ 817    0]
 [   0 1129]]
SVM Classification Report:
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       817
           1       1.00      1.00      1.00      1129

    accuracy                           1.00      1946
   macro avg       1.00      1.00      1.00      1946
weighted avg       1.00      1.00      1.00      1946

Confusion Matrix:
 [[ 817    0]
 [   0 1129]]
Naive Bayes Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       817
           1       1.00      1.00      1.00      1129

    accuracy                           1.00      1946
   macro avg       1.00      1.00      1.00      1946
weighted avg       1.00      1.00      1.00      1946

Confusion Matrix:
 [[ 817    0]
 [   0 1129]]
Gradient Boosting Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       817
           1       1.00      1.00      1.00      1129

    accuracy                           1.00      1946
   macro avg       1.00      1.00      1.00      1946
weighted avg       1.00      1.00      1.00      1946

Confusion Matrix:
 [[ 817    0]
 [   0 1129]]
XGBoost Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       817
           1       1.00      1.00      1.00      1129

    accuracy                           1.00      1946
   macro avg       1.00      1.00      1.00      1946
weighted avg       1.00      1.00      1.00      1946
```

```
Confusion Matrix:
 [[ 817    0]
 [   0 1129]]
[LightGBM] [Info] Number of positive: 4490, number of negative: 3292
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of
testing was 0.002418 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 485
[LightGBM] [Info] Number of data points in the train set: 7782, number of used
features: 26
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.576973 -> initscore=0.310357
[LightGBM] [Info] Start training from score 0.310357
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

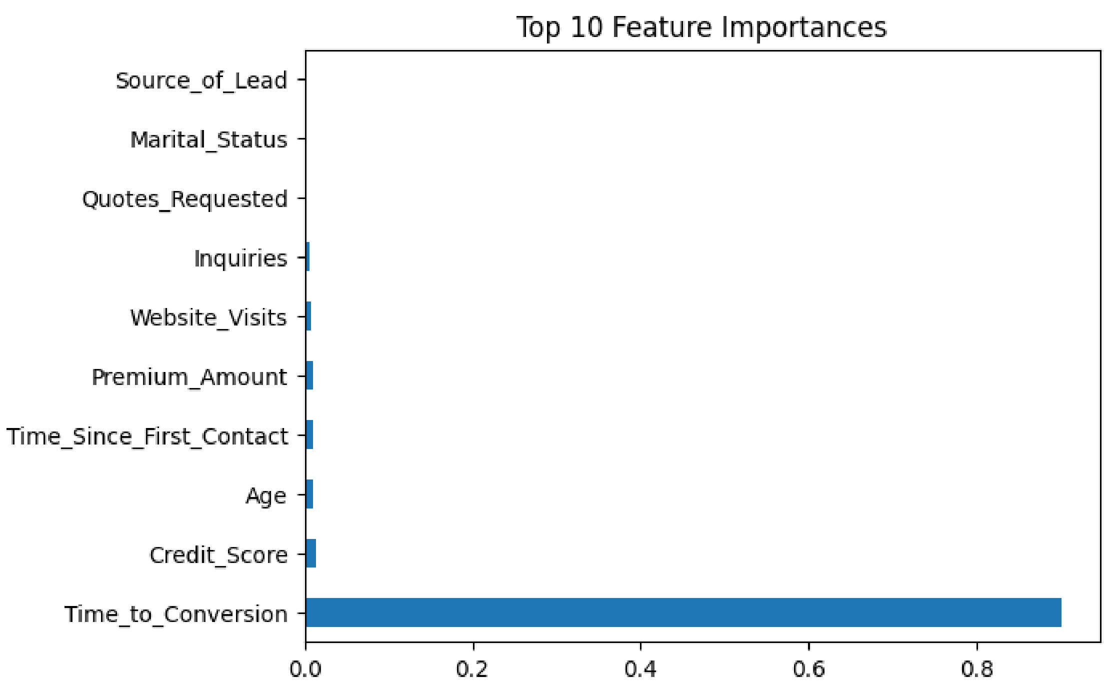
```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
LightGBM Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       817
           1       1.00      1.00      1.00      1129

    accuracy                           1.00      1946
   macro avg       1.00      1.00      1.00      1946
weighted avg       1.00      1.00      1.00      1946

Confusion Matrix:
 [[ 817    0]
 [   0 1129]]
CatBoost Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       817
           1       1.00      1.00      1.00      1129

    accuracy                           1.00      1946
   macro avg       1.00      1.00      1.00      1946
weighted avg       1.00      1.00      1.00      1946

Confusion Matrix:
 [[ 817    0]
 [   0 1129]]
```

[30]:
```python
# Model comparison
results_df = pd.DataFrame(results).T
print(results_df.sort_values(by='AUC', ascending=False))
```

```
                     Accuracy       AUC
Logistic Regression  1.000000  1.000000
Decision Tree        1.000000  1.000000
Gradient Boosting    1.000000  1.000000
Random Forest        1.000000  1.000000
SVM                  1.000000  1.000000
Naive Bayes          1.000000  1.000000
LightGBM             1.000000  1.000000
XGBoost              1.000000  1.000000
CatBoost             1.000000  1.000000
KNN                  0.964029  0.989762
```

[31]:

```python
# Feature importance (Random Forest example)
rfc = RandomForestClassifier()
rfc.fit(X_train, y_train)
feature_importances = pd.Series(rfc.feature_importances_, index=df.
 ↪drop(columns=['Conversion_Status']).columns)
feature_importances.nlargest(10).plot(kind='barh')
plt.title("Top 10 Feature Importances")
plt.show()
```



0.7 About the Author

Name: Arif Mia

Profession: Machine Learning Engineer & Data Scientist

0.7.1 Career Objective

My goal is to contribute to groundbreaking advancements in artificial intelligence and data science, empowering companies and individuals with data-driven solutions. I strive to simplify complex challenges, craft innovative projects, and pave the way for a smarter and more connected future.

As a **Machine Learning Engineer** and **Data Scientist**, I am passionate about using machine learning, deep learning, computer vision, and advanced analytics to solve real-world problems. My expertise lies in delivering impactful solutions by leveraging cutting-edge technologies.

0.7.2 Skills

- **Artificial Intelligence & Machine Learning**
- **Computer Vision & Predictive Analytics**
- **Deep Learning & Natural Language Processing (NLP)**
- **Python Programming & Automation**
- **Data Visualization & Analysis**
- **End-to-End Model Development & Deployment**

0.7.3 Featured Projects

Lung Cancer Prediction with Deep Learning
Achieved 99% accuracy in a computer vision project using 12,000 medical images across three classes. This project involved data preprocessing, visualization, and model training to detect cancer effectively.

Ghana Crop Disease Detection Challenge
Developed a model using annotated images to identify crop diseases with bounding boxes, addressing real-world agricultural challenges and disease mitigation.

Global Plastic Waste Analysis
Utilized GeoPandas, Matplotlib, and machine learning models like RandomForestClassifier and CatBoostClassifier to analyze trends in plastic waste management.

Twitter Emotion Classification
Performed exploratory data analysis and built a hybrid machine learning model to classify Twitter sentiments, leveraging text data preprocessing and visualization techniques.

0.7.4 Technical Skills

- **Programming Languages:** Python , SQL , R
- **Data Visualization Tools:** Matplotlib , Seaborn , Tableau , Power BI
- **Machine Learning & Deep Learning:** Scikit-learn , TensorFlow , PyTorch
- **Big Data Technologies:** Hadoop , Spark
- **Model Deployment:** Flask , FastAPI , Docker

0.7.5 Connect with Me

Email: arifmiahcse@gmail.com

LinkedIn: www.linkedin.com/in/arif-miah-8751bb217

GitHub: https://github.com/Arif-miad

Kaggle: https://www.kaggle.com/arifmia

Let's turn ideas into reality! If you're looking for innovative solutions or need collaboration on exciting projects, feel free to reach out.

How does this look? Feel free to suggest changes or updates!