```
from google.colab import files
uploaded = files.upload()
```

Choose Files WA_Fn-Us…-Attrition.csv

- **WA_Fn-UseC_-HR-Employee-Attrition.csv**(text/csv) - 227977 bytes, last modified: 9/20/2019 - 100% done

```
Saving WA_Fn-UseC_-HR-Employee-Attrition.csv to WA_Fn-UseC_-HR-Employee-Attrition.csv
```

```
import pandas as pd

# Load the dataset
df = pd.read_csv("WA_Fn-UseC_-HR-Employee-Attrition.csv")

# View shape and first few rows
print("Dataset shape:", df.shape)
df.head()
```

```
Dataset shape: (1470, 35)
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	...	RelationshipSatisfaction	Standard
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	...	1	
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	...	4	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	...	2	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	...	3	
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	...	4	

5 rows × 35 columns

```
# Overview of dataset
df.info()

# Check for missing values
df.isnull().sum()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   Age                       1470 non-null   int64 
 1   Attrition                 1470 non-null   object
 2   BusinessTravel            1470 non-null   object
 3   DailyRate                 1470 non-null   int64 
 4   Department                1470 non-null   object
 5   DistanceFromHome          1470 non-null   int64 
 6   Education                 1470 non-null   int64 
 7   EducationField            1470 non-null   object
 8   EmployeeCount             1470 non-null   int64 
 9   EmployeeNumber            1470 non-null   int64 
 10  EnvironmentSatisfaction   1470 non-null   int64 
 11  Gender                    1470 non-null   object
 12  HourlyRate                1470 non-null   int64 
 13  JobInvolvement            1470 non-null   int64 
 14  JobLevel                  1470 non-null   int64 
 15  JobRole                   1470 non-null   object
 16  JobSatisfaction           1470 non-null   int64 
 17  MaritalStatus             1470 non-null   object
 18  MonthlyIncome             1470 non-null   int64 
 19  MonthlyRate               1470 non-null   int64 
 20  NumCompaniesWorked        1470 non-null   int64 
 21  Over18                    1470 non-null   object
 22  OverTime                  1470 non-null   object
 23  PercentSalaryHike         1470 non-null   int64 
 24  PerformanceRating         1470 non-null   int64 
 25  RelationshipSatisfaction  1470 non-null   int64 
 26  StandardHours             1470 non-null   int64 
 27  StockOptionLevel          1470 non-null   int64 
 28  TotalWorkingYears         1470 non-null   int64 
 29  TrainingTimesLastYear     1470 non-null   int64 
 30  WorkLifeBalance           1470 non-null   int64 
 31  YearsAtCompany            1470 non-null   int64 
 32  YearsInCurrentRole        1470 non-null   int64 
 33  YearsSinceLastPromotion   1470 non-null   int64 
 34  YearsWithCurrManager      1470 non-null   int64 
dtypes: int64(26), object(9)
memory usage: 402.1+ KB
```

	0
Age	0
Attrition	0
BusinessTravel	0
DailyRate	0
Department	0
DistanceFromHome	0

Education	0
EducationField	0
EmployeeCount	0
EmployeeNumber	0
EnvironmentSatisfaction	0
Gender	0
HourlyRate	0
JobInvolvement	0
JobLevel	0
JobRole	0
JobSatisfaction	0
MaritalStatus	0
MonthlyIncome	0
MonthlyRate	0
NumCompaniesWorked	0
Over18	0
OverTime	0
PercentSalaryHike	0
PerformanceRating	0
RelationshipSatisfaction	0
StandardHours	0
StockOptionLevel	0
TotalWorkingYears	0
TrainingTimesLastYear	0
WorkLifeBalance	0
YearsAtCompany	0
YearsInCurrentRole	0
YearsSinceLastPromotion	0
YearsWithCurrManager	0

dtype: int64

```
# Label encode the target variable
df['Attrition'] = df['Attrition'].map({'Yes': 1, 'No': 0})


# Get all object (text) columns
categorical_cols = df.select_dtypes(include='object').columns
print("Categorical columns:", list(categorical_cols))
```

```
Categorical columns: ['BusinessTravel', 'Department', 'EducationField', 'Gender', 'JobRole', 'MaritalStatus', 'Over18', 'OverTime']
```

```
# One-hot encode all remaining categorical columns
df_encoded = pd.get_dummies(df, drop_first=True)


# Confirm all columns are now numeric
df_encoded.info()

# Preview the encoded dataset
df_encoded.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 48 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   Age                               1470 non-null   int64
 1   Attrition                         1470 non-null   int64
 2   DailyRate                         1470 non-null   int64
 3   DistanceFromHome                  1470 non-null   int64
 4   Education                         1470 non-null   int64
 5   EmployeeCount                     1470 non-null   int64
 6   EmployeeNumber                    1470 non-null   int64
 7   EnvironmentSatisfaction           1470 non-null   int64
 8   HourlyRate                        1470 non-null   int64
 9   JobInvolvement                    1470 non-null   int64
 10  JobLevel                          1470 non-null   int64
 11  JobSatisfaction                   1470 non-null   int64
 12  MonthlyIncome                     1470 non-null   int64
 13  MonthlyRate                       1470 non-null   int64
 14  NumCompaniesWorked                1470 non-null   int64
 15  PercentSalaryHike                 1470 non-null   int64
 16  PerformanceRating                 1470 non-null   int64
 17  RelationshipSatisfaction          1470 non-null   int64
 18  StandardHours                     1470 non-null   int64
 19  StockOptionLevel                  1470 non-null   int64
 20  TotalWorkingYears                 1470 non-null   int64
 21  TrainingTimesLastYear             1470 non-null   int64
 22  WorkLifeBalance                   1470 non-null   int64
 23  YearsAtCompany                    1470 non-null   int64
 24  YearsInCurrentRole                1470 non-null   int64
 25  YearsSinceLastPromotion           1470 non-null   int64
 26  YearsWithCurrManager              1470 non-null   int64
 27  BusinessTravel_Travel_Frequently  1470 non-null   bool
 28  BusinessTravel_Travel_Rarely      1470 non-null   bool
 29  Department_Research & Development 1470 non-null   bool
 30  Department_Sales                  1470 non-null   bool
 31  EducationField_Life Sciences      1470 non-null   bool
 32  EducationField_Marketing          1470 non-null   bool
 33  EducationField_Medical            1470 non-null   bool
 34  EducationField_Other              1470 non-null   bool
 35  EducationField_Technical Degree   1470 non-null   bool
 36  Gender_Male                       1470 non-null   bool
 37  JobRole_Human Resources           1470 non-null   bool
 38  JobRole_Laboratory Technician     1470 non-null   bool
 39  JobRole_Manager                   1470 non-null   bool
 40  JobRole_Manufacturing Director    1470 non-null   bool
 41  JobRole_Research Director         1470 non-null   bool
 42  JobRole_Research Scientist        1470 non-null   bool
 43  JobRole_Sales Executive           1470 non-null   bool
 44  JobRole_Sales Representative      1470 non-null   bool
 45  MaritalStatus_Married             1470 non-null   bool
 46  MaritalStatus_Single              1470 non-null   bool
 47  OverTime_Yes                      1470 non-null   bool
dtypes: bool(21), int64(27)
memory usage: 340.4 KB
```

	Age	Attrition	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	HourlyRate	JobInvolvement	...	JobRole_Laboratory Technician	JobR
0	41	1	1102	1	2	1	1	2	94	3	...	False	
1	49	0	279	8	1	1	2	3	61	2	...	False	
2	37	1	1373	2	2	1	4	4	92	2	...	True	
3	33	0	1392	3	4	1	5	4	56	3	...	False	
4	27	0	591	2	1	1	7	1	40	3	...	True	

5 rows × 48 columns

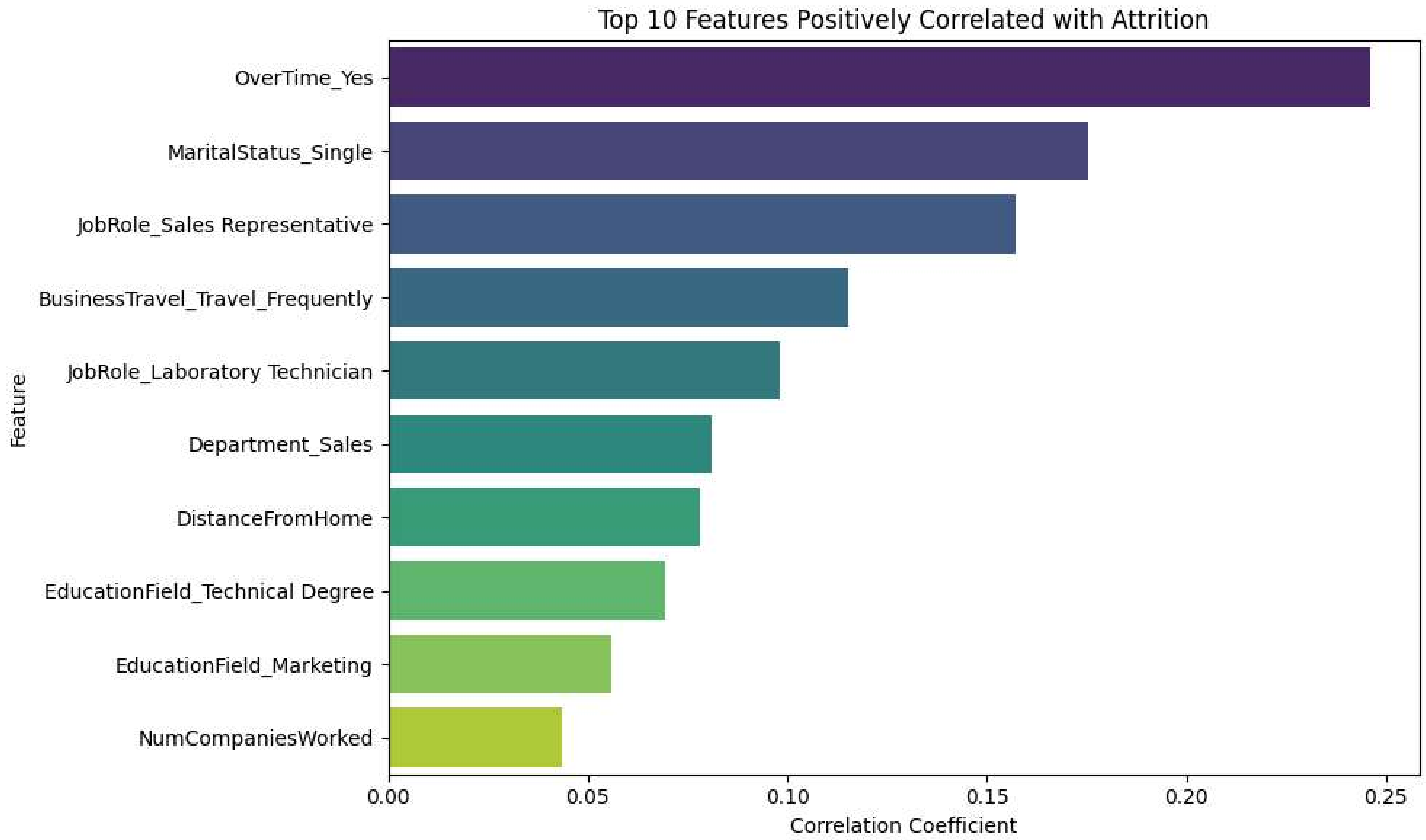
```
# Check correlation with 'Attrition'
corr = df_encoded.corr()['Attrition'].sort_values(ascending=False)
print(corr)
```

```
Attrition                            1.000000
OverTime_Yes                         0.246118
MaritalStatus_Single                 0.175419
JobRole_Sales Representative         0.157234
BusinessTravel_Travel_Frequently     0.115143
JobRole_Laboratory Technician        0.098290
Department_Sales                     0.080855
DistanceFromHome                     0.077924
EducationField_Technical Degree      0.069355
EducationField_Marketing             0.055781
NumCompaniesWorked                   0.043494
JobRole_Human Resources              0.036215
Gender_Male                          0.029453
JobRole_Sales Executive              0.019774
MonthlyRate                          0.015170
PerformanceRating                    0.002889
JobRole_Research Scientist          -0.000360
HourlyRate                          -0.006846
EmployeeNumber                      -0.010577
PercentSalaryHike                   -0.013478
EducationField_Other                -0.017898
Education                           -0.031373
EducationField_Life Sciences        -0.032703
YearsSinceLastPromotion             -0.033019
RelationshipSatisfaction            -0.045872
EducationField_Medical              -0.046999
BusinessTravel_Travel_Rarely        -0.049538
DailyRate                           -0.056652
TrainingTimesLastYear               -0.059478
WorkLifeBalance                     -0.063939
```

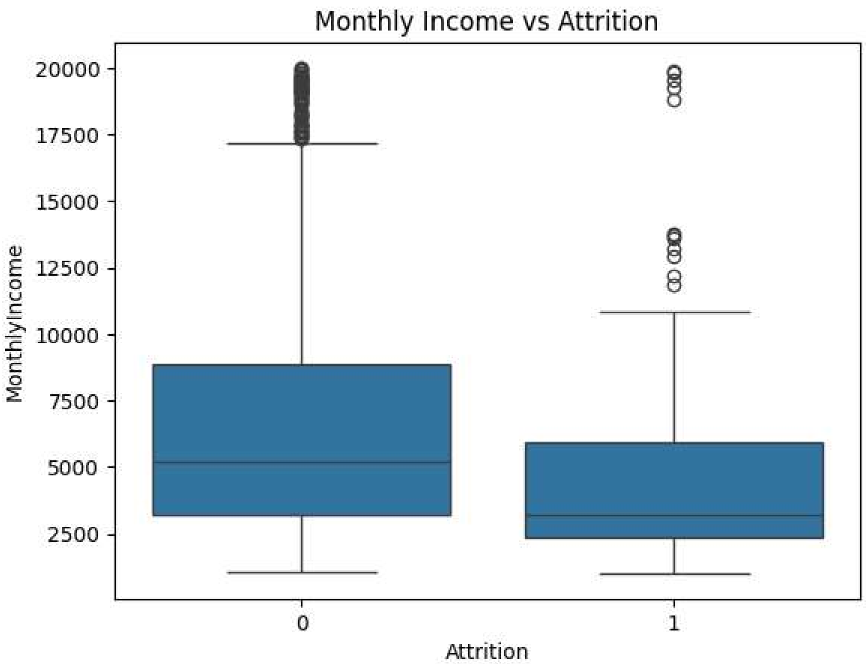
```
JobRole_Manufacturing Director      -0.082994
JobRole_Manager                     -0.083316
Department_Research & Development   -0.085293
JobRole_Research Director           -0.088870
MaritalStatus_Married               -0.090984
EnvironmentSatisfaction             -0.103369
JobSatisfaction                     -0.103481
JobInvolvement                      -0.130016
YearsAtCompany                      -0.134392
StockOptionLevel                    -0.137145
YearsWithCurrManager                -0.156199
Age                                 -0.159205
MonthlyIncome                       -0.159840
YearsInCurrentRole                  -0.160545
JobLevel                            -0.169105
TotalWorkingYears                   -0.171063
EmployeeCount                             NaN
StandardHours                             NaN
Name: Attrition, dtype: float64
```

```python
import matplotlib.pyplot as plt
import seaborn as sns

# Drop Attrition itself and get top correlations
top_corr_features = corr.drop('Attrition').head(10)

plt.figure(figsize=(10, 6))
sns.barplot(x=top_corr_features.values, y=top_corr_features.index, palette="viridis")
plt.title("Top 10 Features Positively Correlated with Attrition")
plt.xlabel("Correlation Coefficient")
plt.ylabel("Feature")
plt.tight_layout()
plt.show()
```

```
<ipython-input-9-9e38d110ffad>:8: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=top_corr_features.values, y=top_corr_features.index, palette="viridis")
```




```
# Example: MonthlyIncome vs Attrition
sns.boxplot(x='Attrition', y='MonthlyIncome', data=df)
plt.title("Monthly Income vs Attrition")
plt.show()
```




```
# Define X and y
X = df_encoded.drop('Attrition', axis=1)
y = df_encoded['Attrition']
```

```
from sklearn.model_selection import train_test_split

# 80/20 split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Initialize models
dt = DecisionTreeClassifier(random_state=42)
rf = RandomForestClassifier(random_state=42)

# Train models
dt.fit(X_train, y_train)
rf.fit(X_train, y_train)
```

```
▾ RandomForestClassifier
RandomForestClassifier(random_state=42)
```

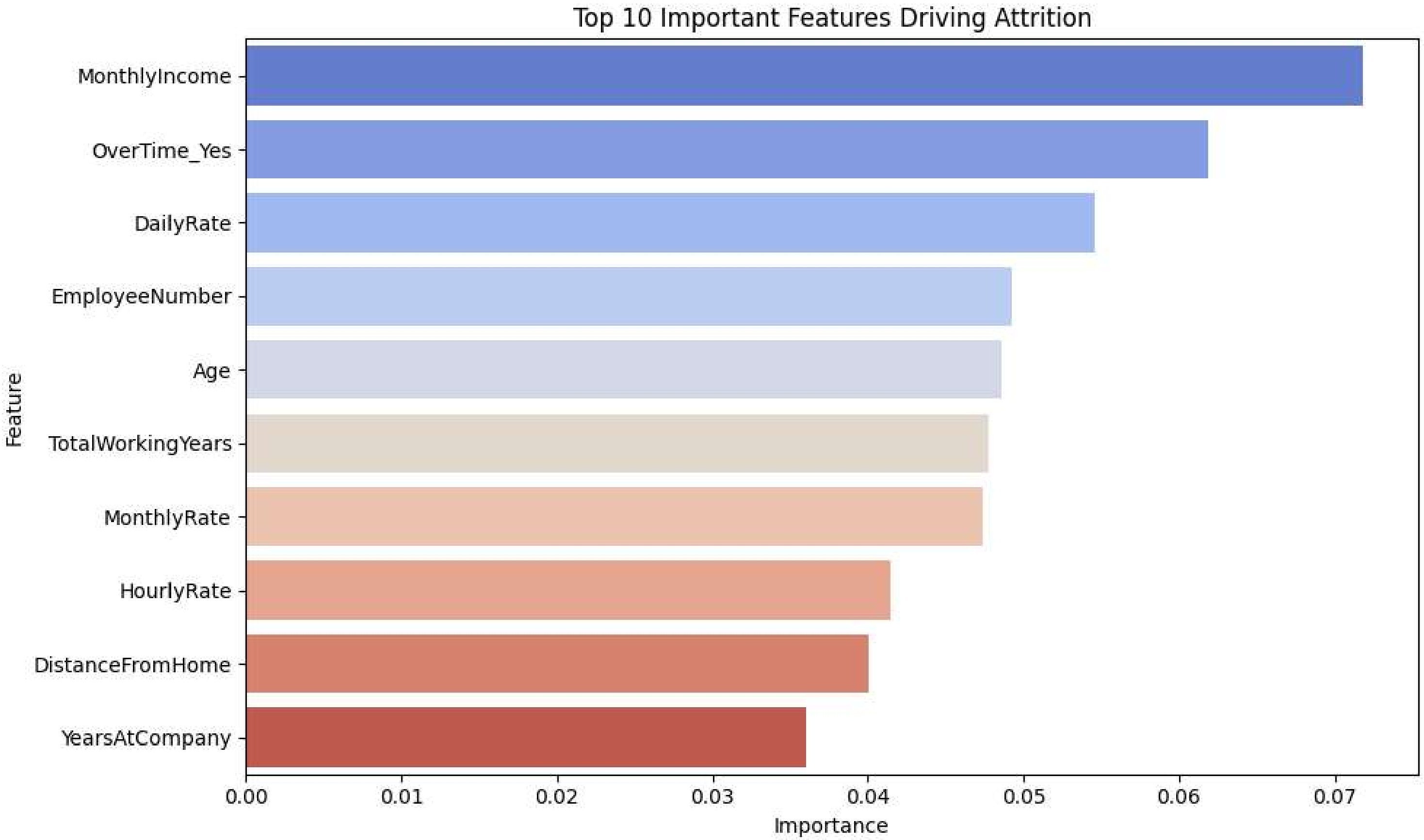
```
# Predict
dt_pred = dt.predict(X_test)
rf_pred = rf.predict(X_test)

# Define a function for evaluation
def evaluate_model(name, y_true, y_pred):
    print(f"\n{name} Results:")
    print("Accuracy:", accuracy_score(y_true, y_pred))
    print("Precision:", precision_score(y_true, y_pred))
    print("Recall:", recall_score(y_true, y_pred))
    print("F1 Score:", f1_score(y_true, y_pred))

# Evaluate both
evaluate_model("Decision Tree", y_test, dt_pred)
evaluate_model("Random Forest", y_test, rf_pred)
```

```
Decision Tree Results:
Accuracy: 0.7755102040816326
Precision: 0.17073170731707318
Recall: 0.1794871794871795
F1 Score: 0.175

Random Forest Results:
Accuracy: 0.8775510204081632
Precision: 0.8
Recall: 0.10256410256410256
F1 Score: 0.18181818181818182
```

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Get feature importances
importances = rf.feature_importances_
features = X.columns

# Create a DataFrame
feat_df = pd.DataFrame({'Feature': features, 'Importance': importances})
feat_df = feat_df.sort_values(by='Importance', ascending=False)


# Plot top 10 important features
plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=feat_df.head(10), palette="coolwarm")
plt.title('Top 10 Important Features Driving Attrition')
```

```
plt.tight_layout()
plt.show()
```

```
<ipython-input-16-d6fcb16262f4>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x='Importance', y='Feature', data=feat_df.head(10), palette="coolwarm")
```




```
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import GradientBoostingClassifier
import xgboost as xgb
from sklearn.ensemble import VotingClassifier

# Initialize models
lr = LogisticRegression(max_iter=1000)
knn = KNeighborsClassifier()
```

```
gb = GradientBoostingClassifier()
xgb_model = xgb.XGBClassifier(use_label_encoder=False, eval_metric='logloss')

# Voting classifier (ensemble of top models)
voting_clf = VotingClassifier(estimators=[
    ('lr', lr), ('knn', knn), ('rf', rf), ('gb', gb), ('xgb', xgb_model)
], voting='soft')


models = {
    "Logistic Regression": lr,
    "K-Nearest Neighbors": knn,
    "Gradient Boosting": gb,
    "XGBoost": xgb_model,
    "Voting Classifier": voting_clf
}

from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print(f"\n{name}")
    print("Accuracy:", accuracy_score(y_test, y_pred))
    print("Precision:", precision_score(y_test, y_pred))
    print("Recall:", recall_score(y_test, y_pred))
    print("F1 Score:", f1_score(y_test, y_pred))
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.py:465: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(

Logistic Regression
Accuracy: 0.8673469387755102
Precision: 0.5
Recall: 0.1794871794871795
F1 Score: 0.2641509433962264

K-Nearest Neighbors
Accuracy: 0.8537414965986394
Precision: 0.35714285714285715
Recall: 0.1282051282051282
F1 Score: 0.18867924528301888

Gradient Boosting
Accuracy: 0.8775510204081632
Precision: 0.6
Recall: 0.23076923076923078
```