```
from google.colab import files

uploaded = files.upload()
```

Choose Files Churn_Modelling.csv

- Churn_Modelling.csv(text/csv) - 684858 bytes, last modified: 3/19/2022 - 100% done

```
Saving Churn_Modelling.csv to Churn_Modelling.csv
```

```
import pandas as pd
data = pd.read_csv('Churn_Modelling.csv')

# Display the first few rows of the DataFrame to ensure it's loaded correctly
data.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMemb
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	

Next steps: Generate code with data | View recommended plots | New interactive sheet

```
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 LogisticRegression
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.metrics import classification_report, roc_auc_score, confusion_matrix, roc_curve
```

```
# Drop unnecessary columns
data = data.drop(['RowNumber', 'CustomerId', 'Surname'], axis=1)
```

```
# Split features and target
X = data.drop('Exited', axis=1)
y = data['Exited']
```

```
# Define categorical and numerical columns
categorical_cols = ['Geography', 'Gender']
numerical_cols = ['CreditScore', 'Age', 'Tenure', 'Balance', 'NumOfProducts',
                  'HasCrCard', 'IsActiveMember', 'EstimatedSalary']
```

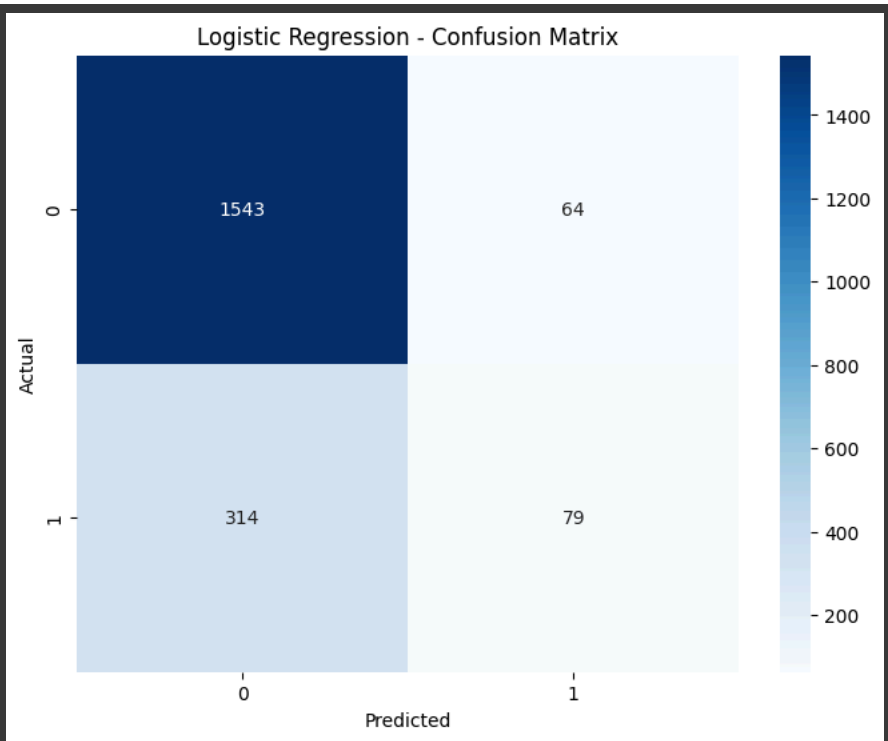
```
# Preprocessing pipeline
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_cols),
        ('cat', OneHotEncoder(), categorical_cols)])
```

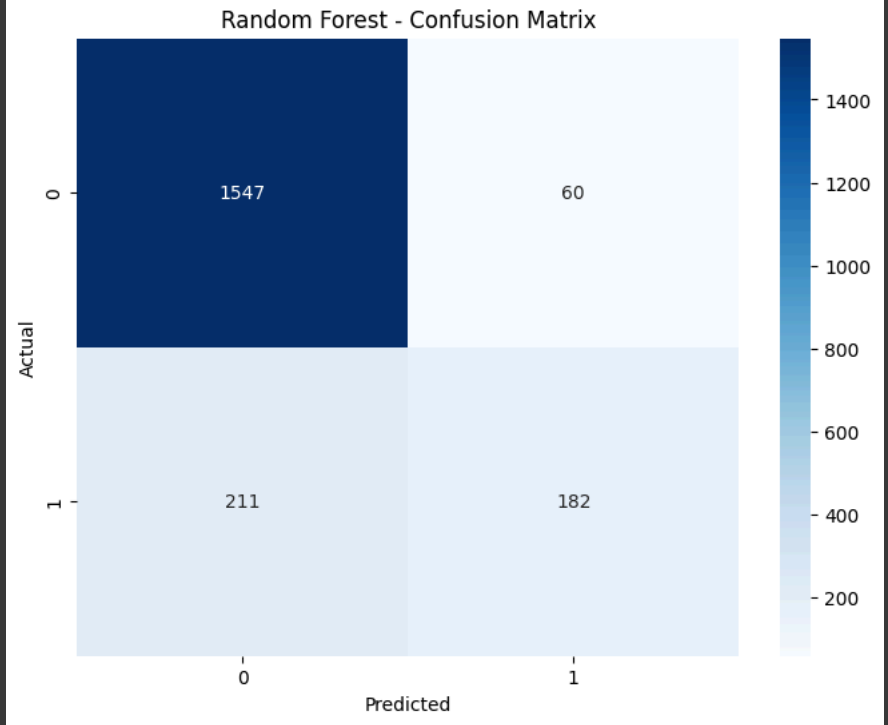
```
# Logistic Regression pipeline
log_reg_pipeline = Pipeline(steps=[('preprocessor', preprocessor),
                                   ('classifier', LogisticRegression())])

# Random Forest pipeline
rf_pipeline = Pipeline(steps=[('preprocessor', preprocessor),
                              ('classifier', RandomForestClassifier(random_state=42))])

# Gradient Boosting pipeline
gb_pipeline = Pipeline(steps=[('preprocessor', preprocessor),
                              ('classifier', GradientBoostingClassifier(random_state=42))])
```

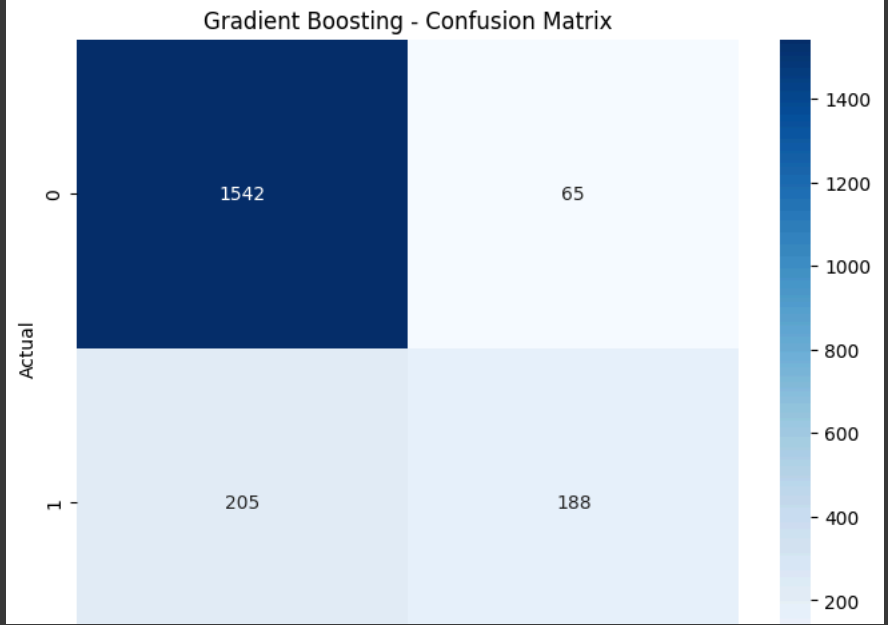
```
# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

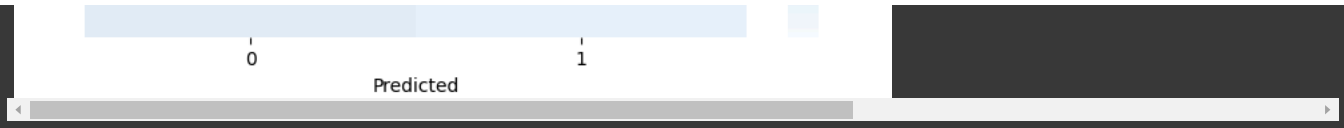
```
# Train models
pipelines = {}
for model_name, model in models.items():
    pipeline = Pipeline(steps=[('preprocessor', preprocessor), ('classifier', model)])
    pipeline.fit(X_train, y_train)
    pipelines[model_name] = pipeline
```

```
for model_name, pipeline in pipelines.items():
    y_pred = pipeline.predict(X_test)
    cm = confusion_matrix(y_test, y_pred)
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.title(f'{model_name} - Confusion Matrix')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.show()
```

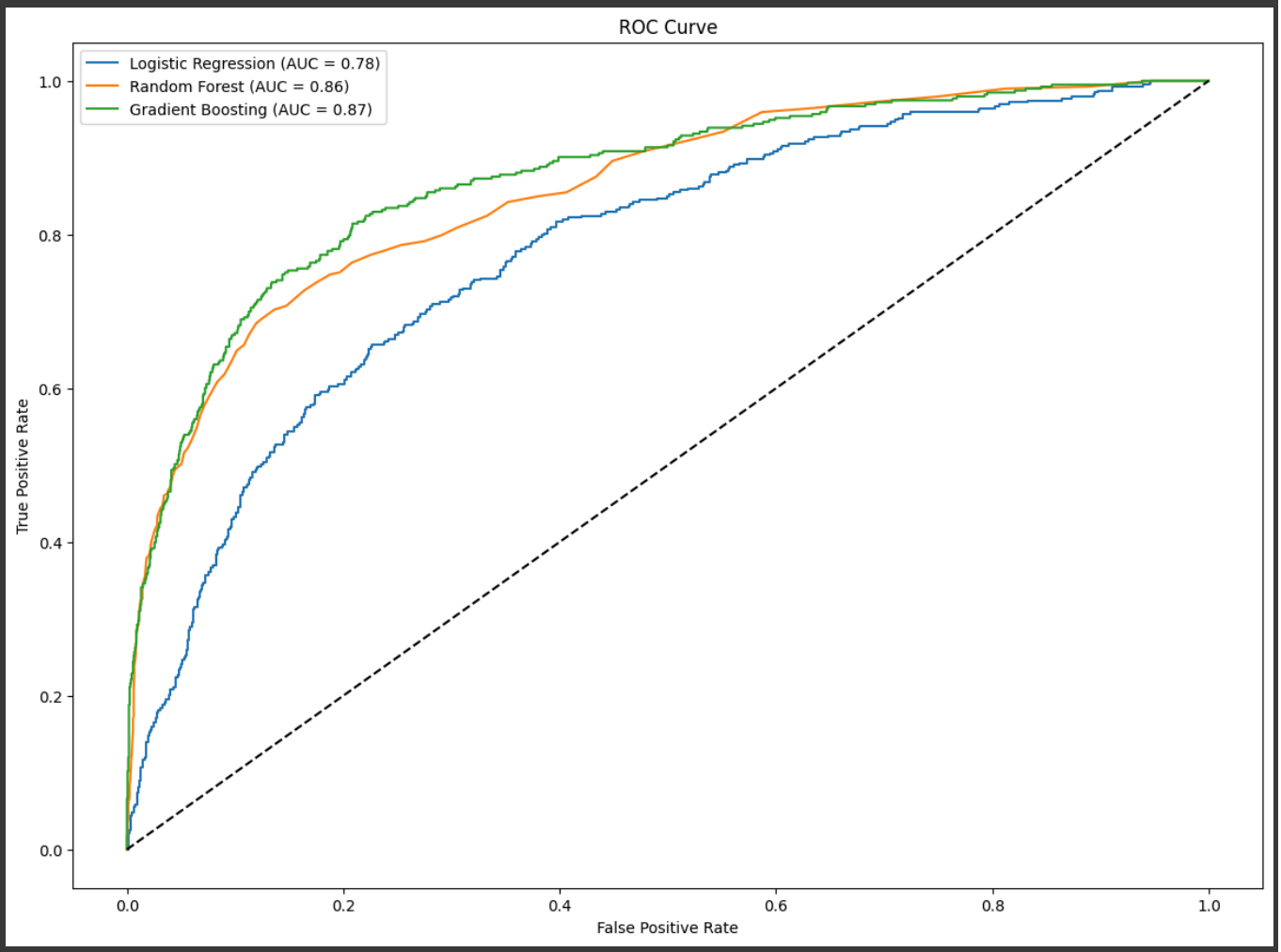

Logistic Regression - Confusion Matrix
1543
64
314
79
Actual
Predicted
0
1
1400
1200
1000
800
600
400
200


Random Forest - Confusion Matrix
1547
60
211
182
Actual
Predicted
0
1
1400
1200
1000
800
600
400
200

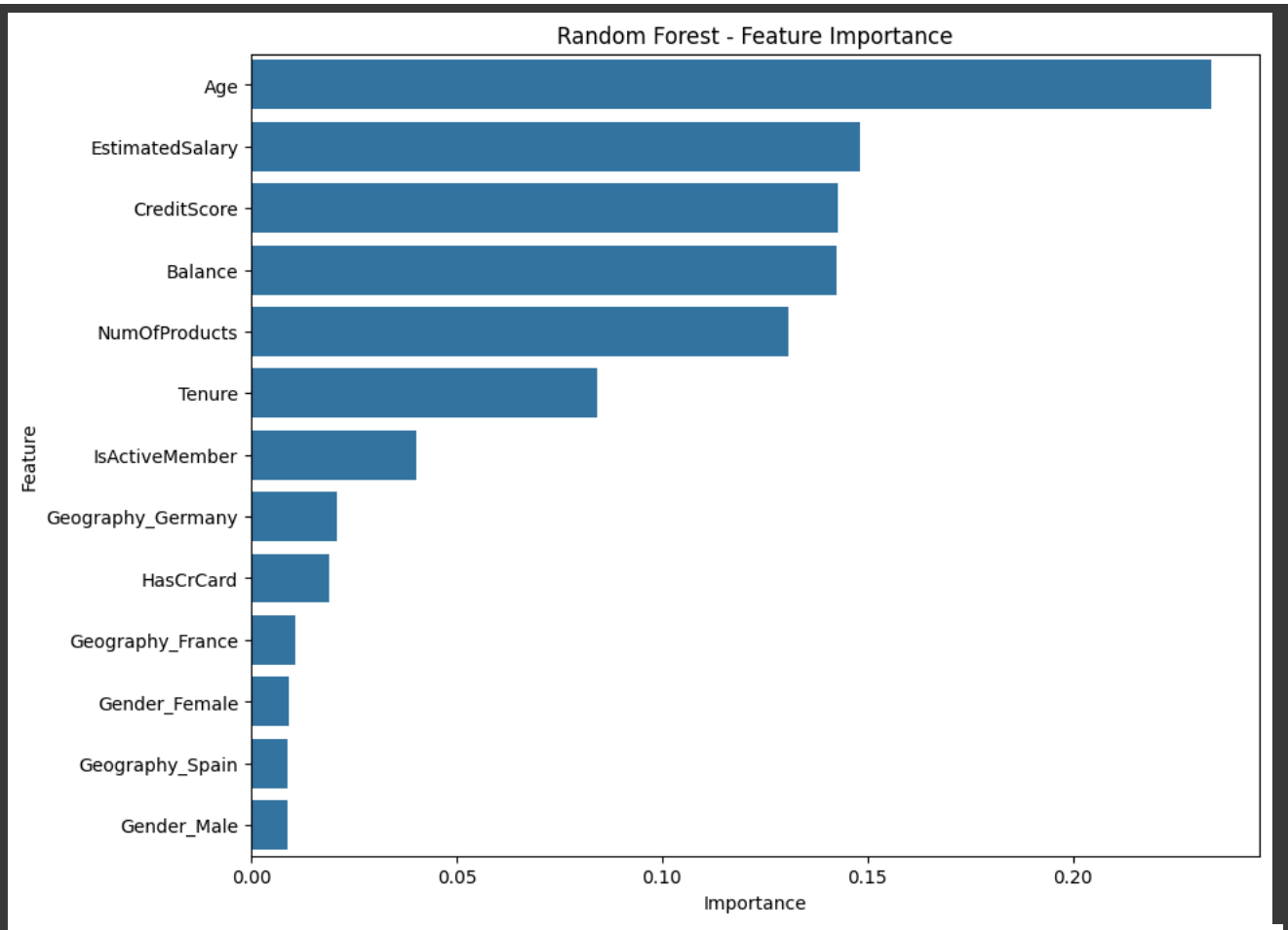

Gradient Boosting - Confusion Matrix
1542
65
205
188
Actual
0
1
1400
1200
1000
800
600
400
200

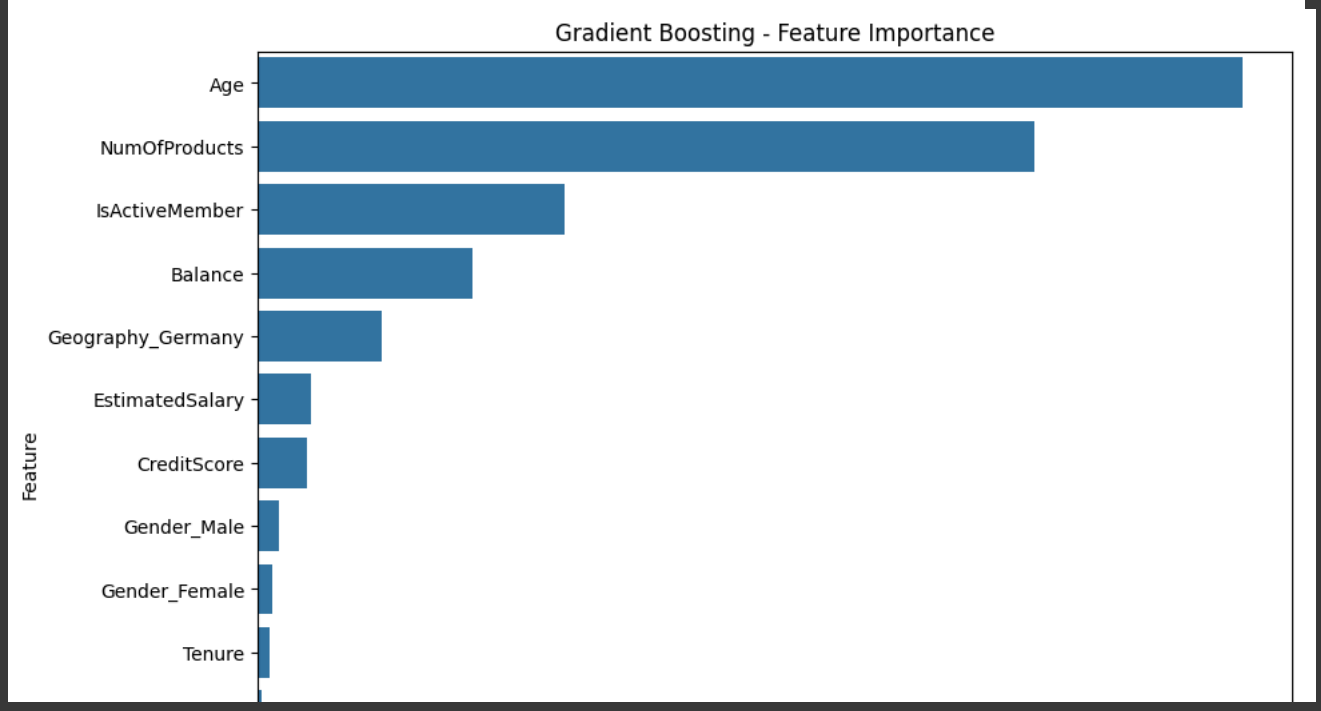



```
plt.figure(figsize=(14, 10))

for model_name, pipeline in pipelines.items():
    y_pred_prob = pipeline.predict_proba(X_test)[:, 1]
    fpr, tpr, _ = roc_curve(y_test, y_pred_prob)
    plt.plot(fpr, tpr, label=f'{model_name} (AUC = {roc_auc_score(y_test, y_pred_prob):.2f})')

plt.plot([0, 1], [0, 1], 'k--')  # Diagonal line
plt.title('ROC Curve')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc='best')
plt.show()
```




```
for model_name, pipeline in pipelines.items():
    if model_name in ['Random Forest', 'Gradient Boosting']:
        model = pipeline.named_steps['classifier']
        feature_importance = model.feature_importances_
        feature_names = numerical_cols + list(pipeline.named_steps['preprocessor'].transformers_[1][1].get_feature_names_out(ca
        importance_df = pd.DataFrame({'Feature': feature_names, 'Importance': feature_importance})
        importance_df = importance_df.sort_values(by='Importance', ascending=False)

        plt.figure(figsize=(10, 8))
        sns.barplot(x='Importance', y='Feature', data=importance_df)
        plt.title(f'{model_name} - Feature Importance')
        plt.show()
```


Random Forest - Feature Importance
Age
EstimatedSalary
CreditScore
Balance
NumOfProducts
Tenure
IsActiveMember
Geography_Germany
HasCrCard
Geography_France
Gender_Female
Geography_Spain
Gender_Male
Feature
0.00
0.05
0.10
0.15
0.20
Importance


Gradient Boosting - Feature Importance
Age
NumOfProducts
IsActiveMember
Balance
Geography_Germany
EstimatedSalary
CreditScore
Gender_Male
Gender_Female
Tenure
Feature