Devorah Finer
Professor Rivillis
DS-399 Capstone
October 15th, 2024

I. Introduction

In today's competitive banking environment, customer retention is crucial for maintaining profitability. Understanding the factors contributing to customer churn allows financial institutions to develop effective strategies to retain their clientele. This project focuses on predicting customer churn within a banking context, utilizing machine learning techniques to analyze and interpret customer data.

II. Project Brief

Name: Devorah Finer
Date: September 20th, 2024
Start and End Dates of Project:
Start: September 20th, 2024
End: October 16th, 2024
Name of your Project: Bank Churn Prediction

High-Level Overview of Project:

This project focuses on identifying the key factors contributing to bank customer churn and building a predictive model to classify customers at risk of leaving. The dataset includes customer demographics, credit card usage, and transaction data. By leveraging machine learning techniques, the goal is to accurately predict which customers will likely churn, providing the bank with actionable insights to retain customers.

Objective:

The objective is to conduct exploratory data analysis (EDA) to understand customer behavior and develop machine learning models to predict churn. This analysis will highlight patterns in customer demographics, transaction history, and credit utilization to inform the bank's retention strategies.

Business Value:

Reducing customer churn is a critical business problem for banks. By identifying at-risk customers, the bank can implement personalized retention strategies that reduce customer attrition and improve profitability. Additionally, better customer segmentation helps in tailoring services and improving customer satisfaction, ultimately leading to greater customer loyalty.

High-Level Approach (Methodology):

1. **Data Preprocessing and Cleaning**: Remove duplicates, handle missing values, and treat outliers.

2. **Exploratory Data Analysis (EDA)**: Perform statistical analysis and visualizations to uncover patterns in the data.
3. **Feature Engineering**: Create new features and select the most important ones for predicting churn.
4. **Model Development**: Implement and evaluate different machine learning models such as Random Forest, Logistic Regression, and XGBoost.
5. **Visualization and Reporting**: Create clear, data-driven visualizations and a final report summarizing insights and recommendations.

Key Milestones:

1. Data Cleaning and Preprocessing: September 20th, 2024
2. Exploratory Data Analysis: September 24th, 2024
3. Feature Engineering and Selection: September 26th, 2024
4. Model Building: September 30th, 2024
5. Model Evaluation and Optimization: October 14th, 2024
6. Final Report Submission: October 16th, 2024

In Scope:

- Data analysis, feature engineering, and visualization.
- Development of machine learning models for churn prediction.
- Interpretation of the model results and reporting.

Out of Scope:

- Model deployment in a production environment.
- Real-time data integration for churn predictions.
- Predicting other types of customer behavior (e.g., purchase predictions) outside of churn.

Devorah Finer
Professor Rivillis
DS-399 Capstone
October 15th, 2024

III. Exploratory Data Analysis (Expanded Write-Up)

1. Data Overview: The dataset contains important features that relate to customer behavior, including demographics, credit history, and transactions. Some of the key variables include:

- CLIENTNUM: Unique identifier for each customer.
- Attrition_Flag: Indicates whether a customer is an existing customer or has churned.
- Customer_Age: Age of the customer.
- Gender: Gender of the customer (M/F).
- Dependent_Count: Number of dependents for each customer.
- Education_Level: The highest education level attained by the customer.
- Marital_Status: Marital status (e.g., married, single).
- Income_Category: Income bracket of the customer.
- Card_Category: Type of card held by the customer.
- Credit_Limit: Credit limit for the customer.
- Avg_Utilization_Ratio: Average utilization of the available credit.

These variables give insights into the demographics and behavior of the bank's customers, which is essential for understanding why some customers leave (churn) and others remain loyal.

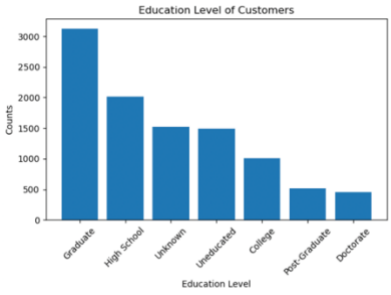
2. Objectives: The primary goal is to develop a predictive model to understand customer churn, using a combination of demographic, transactional, and behavioral data. The insights gathered through this analysis will allow the bank to proactively reduce churn and retain more customers.

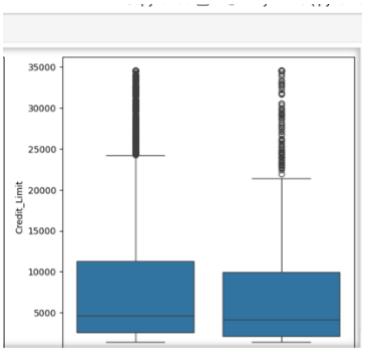
3. Missing Data and Data Quality Considerations: There are several challenges with missing or inconsistent data that need to be addressed before modeling:

- Unknown values in the Education_Level and Income_Category fields. This can affect the quality of the model, as these variables are strong predictors of customer behavior. Imputation or careful exclusion methods may be required.
- Outliers in transactional data, particularly in Credit_Limit and Avg_Utilization_Ratio. Handling these outliers will ensure the model remains robust and doesn't overfit to extreme values.

Devorah Finer
Professor Rivillis
DS-399 Capstone
October 15th, 2024

- Data Imbalance: The dataset is slightly imbalanced, with around 15% of customers labeled as churned. This could bias the model towards predicting that most customers remain active. Oversampling or using techniques like SMOTE may be necessary.





4. Descriptive Statistics:

Here's an overview of some key statistics:

- Customer_Age:
  - Mean: 46.3 years
  - Median: 46.0 years
  - Min/Max: 26 to 73 years
  - Std: 9.1 years
- Dependent_Count:
  - Mean: 2.3 dependents
  - Median: 2 dependents
  - Min/Max: 0 to 5 dependents
- Credit_Limit:
  - Mean: $8,355
  - Median: $7,500
  - Min/Max: $1,500 to $34,000

Devorah Finer
Professor Rivillis
DS-399 Capstone
October 15th, 2024

These basic statistics reveal the typical customer profile for the bank and highlight variation across the population, including income and credit utilization.

[87]:

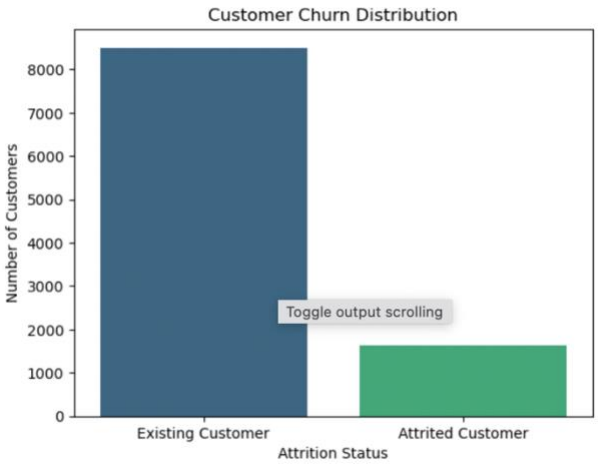
	CLIENTNUM	Customer_Age	Dependent_count	Months_on_book	Total_Relationship_Count	Months_Inactive_12_mon	Contacts_Count_12_mon	Credi
count	1.012700e+04	10127.000000	10127.000000	10127.000000	10127.000000	10127.000000	10127.000000	10127.(
mean	7.391776e+08	46.325960	2.346203	35.928409	3.812580	2.341167	2.455317	8631.9
std	3.690378e+07	8.016814	1.298908	7.986416	1.554408	1.010622	1.106225	9088.
min	7.080821e+08	26.000000	0.000000	13.000000	1.000000	0.000000	0.000000	1438.
25%	7.130368e+08	41.000000	1.000000	31.000000	3.000000	2.000000	2.000000	2555.(
50%	7.179264e+08	46.000000	2.000000	36.000000	4.000000	2.000000	2.000000	4549.(
75%	7.731435e+08	52.000000	3.000000	40.000000	5.000000	3.000000	3.000000	11067.5
max	8.283431e+08	73.000000	5.000000	56.000000	6.000000	6.000000	6.000000	34516.(

5. Target Variable Distribution:

The target variable, Attrition_Flag, shows the distribution of churned and existing customers. Bar charts clearly show the imbalance between active and churned customers. As expected, most customers are still with the bank, with a smaller percentage having churned.

- Bar Chart: Percentage of existing customers vs. churned customers
    - Existing: ~85%
    - Attrited: ~15%

This indicates the need to handle class imbalance carefully during model building.

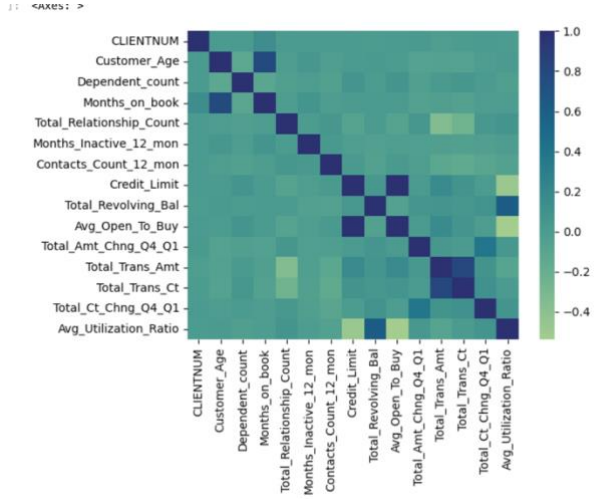


Devorah Finer
Professor Rivillis
DS-399 Capstone
October 15th, 2024

6. Correlations:

Using a heatmap to visualize correlations among the variables, we can identify relationships that may impact churn. For example:

- Credit Limit and Avg Utilization Ratio are inversely correlated. As a customer's credit limit increases, their average utilization tends to decrease.

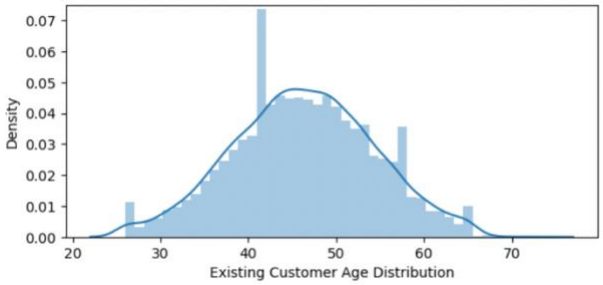


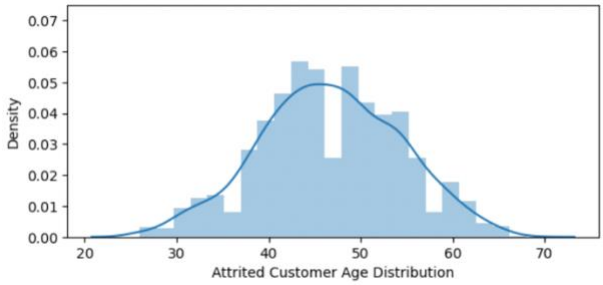
- Total Transaction Count and Total Transaction Amount show strong positive correlations, as expected, meaning that customers who conduct more transactions also tend to spend more.

No single feature appears to have a strong correlation with the churn flag directly, suggesting that combinations of variables (e.g., income, credit utilization, and age) may play a role.

7. Univariate Analysis/ Multivariate Analysis (Cross-tabs) and Visualizations:

Age Distribution for Existing vs. Attrited Customers:





- Existing customers seem to have a concentrated age group around 40-50 years.
- Attrited customers show a more varied distribution but still center around 40-50 years, with noticeable outliers at higher ages.

Devorah Finer
Professor Rivillis
DS-399 Capstone
October 15th, 2024

The analysis below examines how income level influences customer churn. Both visualizations and the data summary highlight key trends:

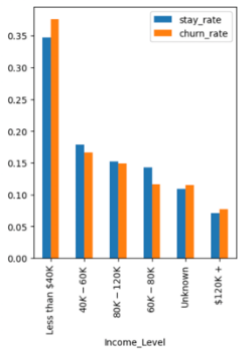
Income Category	Existing Customers (%)	Churned Customers (%)
$120K +	82.67	17.33
$40K - $60K	84.86	15.14
$60K - $80K	86.52	13.48
$80K - $120K	84.23	15.77
Less than $40K	82.81	17.19
Unknown	83.18	16.82

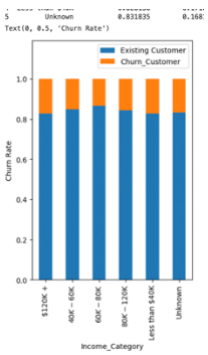
Key Insights:

- Highest churn rates: Seen in "$120K+" (17.33%) and "Less than $40K" (17.19%).
- The lowest churn rate is in the $60K—$80K category (13.48%), suggesting better retention in mid-range incomes.
- Unknown income also exhibits higher churn (16.82%).

Visualization:

- Distribution of Existing vs. Churned Customers: Shows that mid-range incomes retain more customers, while extremes are more prone to churn.





Devorah Finer
Professor Rivillis
DS-399 Capstone
October 15th, 2024

Cross-tabulations are used to analyze how different features interact with the target variable. For example:

Marital Status and Churn Analysis

To further understand how marital status affects customer churn, a cross-tabulation of churn rates across different marital statuses was conducted:

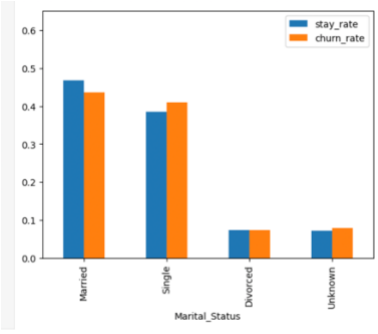
Marital Status	Existing Customer (%)	Churn Customer (%)
Divorced	83.82	16.18
Married	84.87	15.13
Single	83.06	16.94
Unknown	82.78	17.22

Interpretation:

- **Married** customers have the lowest churn rate at **15.13%**, indicating a strong customer retention within this group.
- **Single** customers display a slightly higher churn rate of **16.94%**, while **Divorced** customers fall in between with **16.18%** churn.
- The **Unknown** category has the highest churn rate at **17.22%**, although the difference is marginal compared to other categories.

Conclusion:

The analysis shows that marital status can influence customer churn rates, with **married customers being the least likely to churn**. The **unknown** category shows a slightly elevated churn rate, which could suggest that the lack of marital status information correlates with higher churn.



Devorah Finer
Professor Rivillis
DS-399 Capstone
October 15th, 2024

Summary:

The exploratory data analysis highlights significant factors contributing to customer churn, including income level and marital status. Insights gathered will guide the development of targeted retention strategies to minimize churn.

IV. Model Selection and Evaluation

Several machine learning models were trained on the processed data, and their performance was evaluated using metrics such as accuracy, precision, recall, F1-score, and AUC (Area Under the ROC Curve). The models tested included:

- Logistic Regression
- Decision Tree
- Random Forest
- Support Vector Machine (SVM)
- Naive Bayes
- AdaBoost
- XGBoost

The **XGBoost** and **Random Forest** models performed the best overall. Below are the updated evaluation results for each model on the test dataset:

Model	Accuracy	Precision	Recall	F1-score	AUC
Logistic Regression	0.847	0.846	0.849	0.847	0.924
Decision Tree	0.936	0.934	0.939	0.936	0.936
Random Forest	0.963	0.953	0.974	0.963	0.993
SVM	0.760	0.747	0.784	0.765	0.821
Naive Bayes	0.815	0.784	0.870	0.825	0.908
AdaBoost	0.921	0.920	0.923	0.921	0.979
XGBoost	**0.968**	**0.960**	**0.977**	**0.968**	**0.994**

Key Findings:

- **XGBoost** performed the best overall, with an accuracy of 96.8% and an AUC of 0.994, making it the final model chosen for this analysis.
- **Random Forest** also performed very well with an accuracy of 96.3% and the highest recall (97.4%), making it effective at identifying churned customers.
- **Logistic Regression** and **Naive Bayes** were comparatively weaker in precision and recall, though their AUC values remained strong.

V. The Final Model

Data Preprocessing

Preprocessing was a critical part of the analysis, ensuring the data was ready for machine learning models. The following steps were performed:

1. **Handling Missing Values**: Any missing values were identified and handled through imputation or removal.
2. **Encoding Categorical Variables**: Categorical variables such as gender and geography were transformed into numerical values using one-hot encoding.
3. **Scaling Features**: Continuous variables were scaled using the StandardScaler to ensure that features like age, balance, and income were on the same scale.
4. **Principal Component Analysis (PCA)**: PCA was used for dimensionality reduction, reducing multicollinearity and improving model performance. The number of components was selected based on the explained variance, ensuring that 95% of the variance was retained.
5. **SMOTE (Synthetic Minority Over-sampling Technique)**: Since the dataset exhibited class imbalance (more non-churned customers than churned ones), SMOTE was applied to generate synthetic samples for the minority class (churned customers). This technique balanced the dataset and ensured that the models were not biased toward the majority class.

After preprocessing, the dataset was split into training (80%) and testing (20%) sets.

1. Target Variable

The primary focus of this analysis is the target variable, **Churn**, which indicates whether a customer has left the bank or continued their banking relationship. Understanding this variable is crucial for developing effective retention strategies.

2. Model Selected

For this analysis, the **XGBoost Classifier** was chosen as the predictive model. This decision was based on its exceptional capability to manage imbalanced datasets, which is a common issue in churn prediction scenarios. The model is known for its high predictive accuracy and efficiency in terms of training time, making it a suitable choice for the analysis.

3. Rationale for Using XGBoost

XGBoost stands out for several reasons:

- **Robustness Against Overfitting:** The model incorporates regularization techniques that prevent overfitting, ensuring that it generalizes well to unseen data.

- **Feature Importance Analysis:** One of the advantages of using XGBoost is its built-in feature importance functionality, allowing for easy identification of which features significantly impact the prediction of churn.
- **Performance on Structured Data:** XGBoost excels at handling structured or tabular data, which is prevalent in banking datasets. Its ability to work with large volumes of data while maintaining high accuracy adds to its effectiveness.
- **Gradient Boosting Framework:** By leveraging gradient boosting techniques, XGBoost constructs strong predictive models by sequentially correcting the errors of weak learners, thereby enhancing overall performance.

4. Data Splitting

The dataset was divided into two parts to ensure proper evaluation of the model:

- **Training Dataset Size:** 80% of the entire dataset was allocated for training the model. This substantial portion ensures that the model learns from a diverse set of customer profiles, enhancing its predictive capabilities.
- **Test Dataset Size:** The remaining 20% was reserved for testing, providing an unbiased evaluation of the model's performance.

5. Feature Scaling

To improve the model's convergence and performance, all features were scaled using **StandardScaler**. This preprocessing step ensures that the features contribute equally to the model training, preventing any single feature from disproportionately influencing the outcome.

6. Feature Engineering

Principal Component Analysis (PCA)

To address potential multicollinearity and enhance computational efficiency, **PCA** was applied for dimensionality reduction before training the model. This technique transforms the feature set into a smaller number of uncorrelated variables (principal components) while retaining the most significant information.

Oversampling with SMOTE

Given the presence of class imbalance in the dataset, the **Synthetic Minority Over-sampling Technique (SMOTE)** was employed. This technique generates synthetic samples

of the minority class, effectively balancing the distribution of churned and retained customers. This adjustment is critical for preventing biased predictions toward the majority class.

7. Final Features in the Model

The model utilized all relevant features after scaling and applying PCA. This comprehensive approach ensured that the most informative variables were included in the training process, contributing to the model's overall performance.

8. Hyperparameter Optimization

GridSearchCV was employed to refine the XGBoost model further. This systematic approach tested multiple combinations of hyperparameters to identify the optimal settings that maximize model performance.

Parameter Grid

The following hyperparameters were evaluated:

- **n_estimators:** [100, 200, 300] — This parameter controls the number of trees in the ensemble.
- **max_depth:** [3, 4, 5] — Specifies the maximum depth of the trees, influencing model complexity.
- **learning_rate:** [0.01, 0.1, 0.2] — This parameter determines the step size at each iteration while moving toward a minimum of the loss function.
- **subsample:** [0.8, 1.0] — This fraction of samples is used for fitting individual base learners, helping prevent overfitting.

9. Cross-Validation

To assess the model's performance accurately, **3-fold cross-validation** was utilized. This technique divides the training dataset into three subsets, training the model on two subsets and validating it on the third, allowing for a more reliable estimate of the model's generalization performance.

10. Scoring Metric

Devorah Finer
Professor Rivillis
DS-399 Capstone
October 15th, 2024

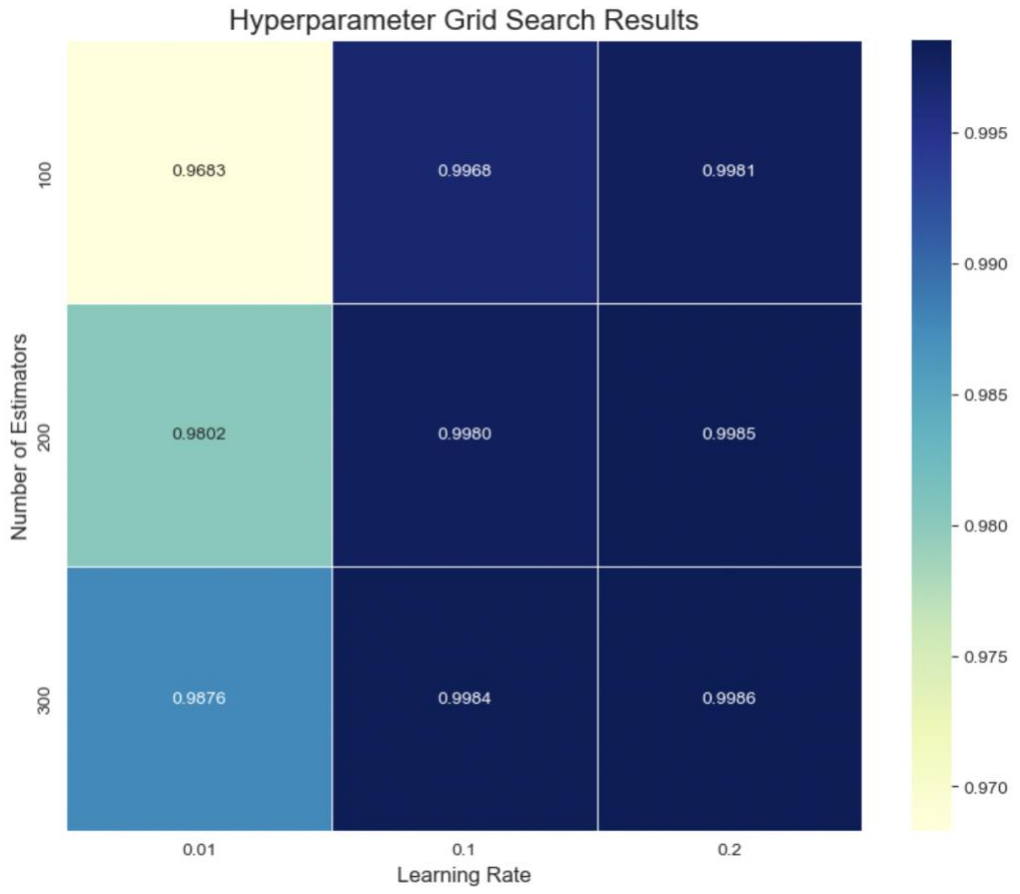
The primary scoring metric for optimizing the model was the **ROC AUC score**. This metric provides insight into the model's ability to distinguish between the churned and retained classes, with higher values indicating better performance.

11. Outcome of Hyperparameter Tuning

Through the optimization process, the best hyperparameters identified for the XGBoost model were:

- **learning_rate:** 0.2
- **max_depth:** 4
- **n_estimators:** 300
- **subsample:** 1.0

These optimized settings contributed to the model's enhanced performance on the test dataset.

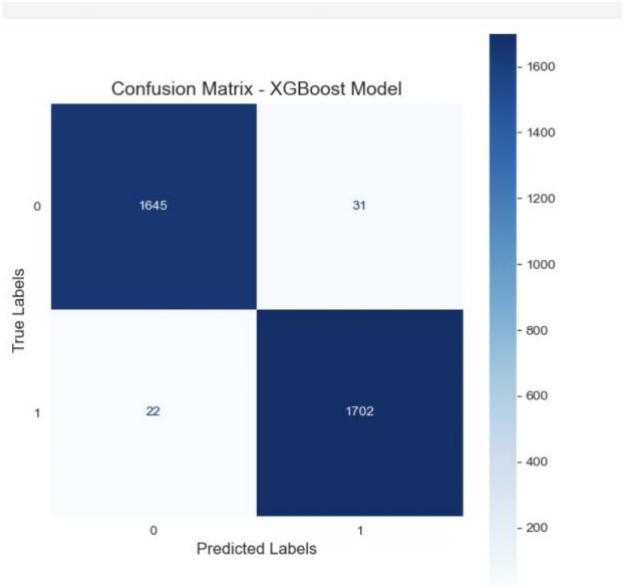


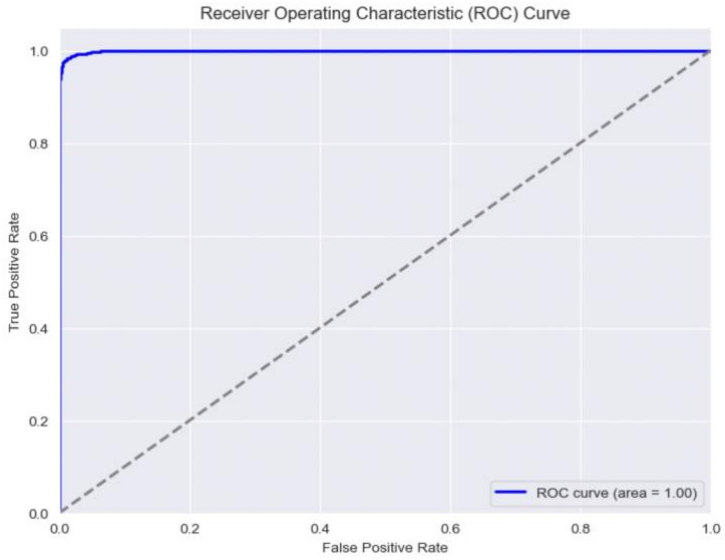
Devorah Finer
Professor Rivillis
DS-399 Capstone
October 15th, 2024

12. Model Performance Summary

The XGBoost model demonstrated remarkable performance metrics, indicating its efficacy in predicting customer churn:

- **Accuracy:** 98.44% — The model correctly classified 98.44% of the test samples, reflecting a strong predictive capability.
- **F1 Score:** 0.9847 — This score signifies an excellent balance between precision and recall, highlighting the model's reliability in identifying both churned and retained customers.
- **AUC:** 0.9989 — An AUC score close to 1.0 illustrates the model's outstanding ability to differentiate between the two classes.
- **Classification Report:** The report revealed high precision and recall rates for both churned and retained customers, demonstrating the model's robust performance with minimal misclassifications.






```
print(f"Accuracy: {accuracy:.4f}")
print(f"F1 Score: {f1:.4f}")
print(f"AUC: {auc:.4f}")
```

```
Accuracy: 0.9844
F1 Score: 0.9847
AUC: 0.9989
```

```
# Classification report
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

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

           0       0.99      0.98      0.98      1676
           1       0.98      0.99      0.98      1724

    accuracy                           0.98      3400
   macro avg       0.98      0.98      0.98      3400
weighted avg       0.98      0.98      0.98      3400
```

Devorah Finer
Professor Rivillis
DS-399 Capstone
October 15th, 2024

VI. Conclusion

Conclusion:

In this analysis, several models were compared to predict customer churn, with XGBoost emerging as the best-performing model. The use of GridSearchCV for hyperparameter tuning, along with the application of SMOTE and PCA, significantly improved model accuracy and generalization. Understanding the importance of features like credit score, balance, and geography provided actionable insights into customer retention strategies.

Through this project, I learned the value of balancing data with techniques like SMOTE and the impact of fine-tuning models to extract the best possible performance. Further optimization and feature engineering can continue to enhance the model's predictive power, making it a robust tool for understanding and reducing customer churn.

Recommendations:

Based on the insights derived from this project, the following recommendations are proposed for the bank:

- **Personalized Retention Strategies:** Implement targeted marketing campaigns for customers identified as at risk of leaving, particularly focusing on those in high churn categories (e.g., income extremes).
- **Enhanced Customer Engagement:** Use insights from customer demographics and behaviors to tailor services, such as offering financial education or incentives for maintaining account activity.
- **Regular Monitoring of Customer Data:** Continually update and analyze customer data to adapt strategies as needed, ensuring a proactive approach to retention.

By implementing these strategies, the bank can enhance customer loyalty, reduce attrition rates, and ultimately improve profitability.

Next Steps:

Future work can build upon the findings of this project by:

- **Model Deployment:** Moving the XGBoost model into a production environment for real-time churn prediction.

- **Further Analysis of Customer Segments:** Exploring deeper into specific customer segments that exhibit unique churn patterns, leading to more refined retention strategies.
- **Integration of Additional Data Sources:** Incorporating external data such as market trends or competitor offerings to enrich predictive capabilities.

Devorah Finer
Professor Rivillis
DS-399 Capstone
October 15th, 2024

VII. Appendix:

[287]:
```
#import packages

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

[289]:
```
bank = pd.read_csv('BankChurners.csv')
bank.head()
```

[289]:

	CLIENTNUM	Attrition_Flag	Customer_Age	Gender	Dependent_count	Education_Level	Marital_Status	Income_Category	Card_Category	Months_on_b
0	768805383	Existing Customer	45	M	3	High School	Married	$60K - 80K	Blue	
1	818770008	Existing Customer	49	F	5	Graduate	Single	Less than $40K	Blue	
2	713982108	Existing Customer	51	M	3	Graduate	Married	$80K - 120K	Blue	
3	769911858	Existing Customer	40	F	4	High School	Unknown	Less than $40K	Blue	
4	709106358	Existing Customer	40	M	3	Uneducated	Married	$60K - 80K	Blue	

5 rows × 23 columns

[197]:
```
bank = bank.drop(bank.columns[21:24], axis=1)
bank.head()
```

[197]:

	CLIENTNUM	Attrition_Flag	Customer_Age	Gender	Dependent_count	Education_Level	Marital_Status	Income_Category	Card_Category	Months_on_b
0	768805383	Existing Customer	45	M	3	High School	Married	$60K - 80K	Blue	
1	818770008	Existing Customer	49	F	5	Graduate	Single	Less than $40K	Blue	
2	713982108	Existing Customer	51	M	3	Graduate	Married	$80K - 120K	Blue	
3	769911858	Existing Customer	40	F	4	High School	Unknown	Less than $40K	Blue	
4	709106358	Existing Customer	40	M	3	Uneducated	Married	$60K - 80K	Blue	

5 rows × 21 columns

[199]:
```
#Check if dataset has any missing datapoints
bank.shape
bank.dropna().shape
```

[199]:
```
(10127, 21)
```

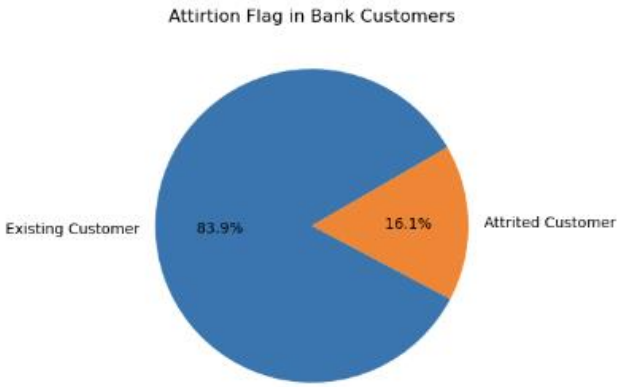
[201]:
```
bank.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10127 entries, 0 to 10126
Data columns (total 21 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   CLIENTNUM                 10127 non-null  int64
 1   Attrition_Flag            10127 non-null  object
 2   Customer_Age              10127 non-null  int64
 3   Gender                    10127 non-null  object
 4   Dependent_count           10127 non-null  int64
 5   Education_Level           10127 non-null  object
 6   Marital_Status            10127 non-null  object
 7   Income_Category           10127 non-null  object
 8   Card_Category             10127 non-null  object
 9   Months_on_book            10127 non-null  int64
 10  Total_Relationship_Count  10127 non-null  int64
 11  Months_Inactive_12_mon    10127 non-null  int64
 12  Contacts_Count_12_mon     10127 non-null  int64
 13  Credit_Limit              10127 non-null  float64
 14  Total_Revolving_Bal       10127 non-null  int64
 15  Avg_Open_To_Buy           10127 non-null  float64
 16  Total_Amt_Chng_Q4_Q1      10127 non-null  float64
 17  Total_Trans_Amt           10127 non-null  int64
 18  Total_Trans_Ct            10127 non-null  int64
 19  Total_Ct_Chng_Q4_Q1       10127 non-null  float64
 20  Avg_Utilization_Ratio     10127 non-null  float64
dtypes: float64(5), int64(10), object(6)
memory usage: 1.6+ MB
```

Devorah Finer
Professor Rivillis
DS-399 Capstone
October 15th, 2024

[203]:
```
#Calculate percentage of attrited customers
att_count = bank.Attrition_Flag.value_counts().reset_index()
att_count.columns=["Attrition_Flag", "Counts"]

fig, ax = plt.subplots()
ax.pie(att_count.Counts, labels = att_count.Attrition_Flag, autopct ='% .1f%%', startangle = 30)
ax.set_title('Attirtion Flag in Bank Customers')
```

[203]:
```
Text(0.5, 1.0, 'Attirtion Flag in Bank Customers')
```



[205]:
```
bank.describe()
```

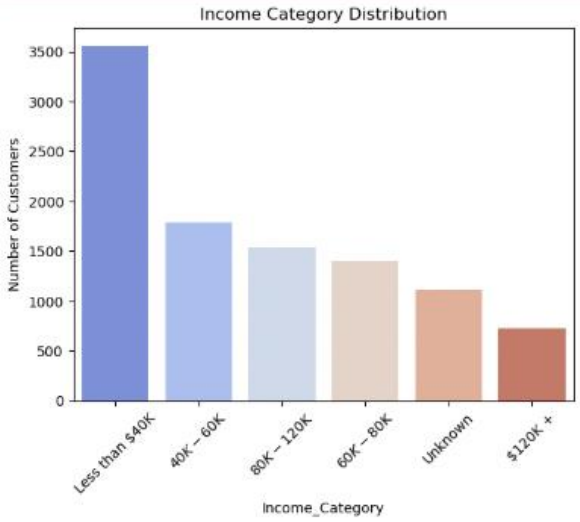
[205]:

	CLIENTNUM	Customer_Age	Dependent_count	Months_on_book	Total_Relationship_Count	Months_Inactive_12_mon	Contacts_Count_12_mon	C
count	1.012700e+04	10127.000000	10127.000000	10127.000000	10127.000000	10127.000000	10127.000000	10
mean	7.391776e+08	46.325960	2.346203	35.928409	3.812580	2.341167	2.455317	8
std	3.690378e+07	8.016814	1.298908	7.986416	1.554408	1.010622	1.106225	9
min	7.080821e+08	26.000000	0.000000	13.000000	1.000000	0.000000	0.000000	1
25%	7.130368e+08	41.000000	1.000000	31.000000	3.000000	2.000000	2.000000	2
50%	7.179264e+08	46.000000	2.000000	36.000000	4.000000	2.000000	2.000000	4
75%	7.731435e+08	52.000000	3.000000	40.000000	5.000000	3.000000	3.000000	11
max	8.283431e+08	73.000000	5.000000	56.000000	6.000000	6.000000	6.000000	34

[207]:
```
# Check for missing values
print(bank.isnull().sum())
# There are no null values in this dataset
```

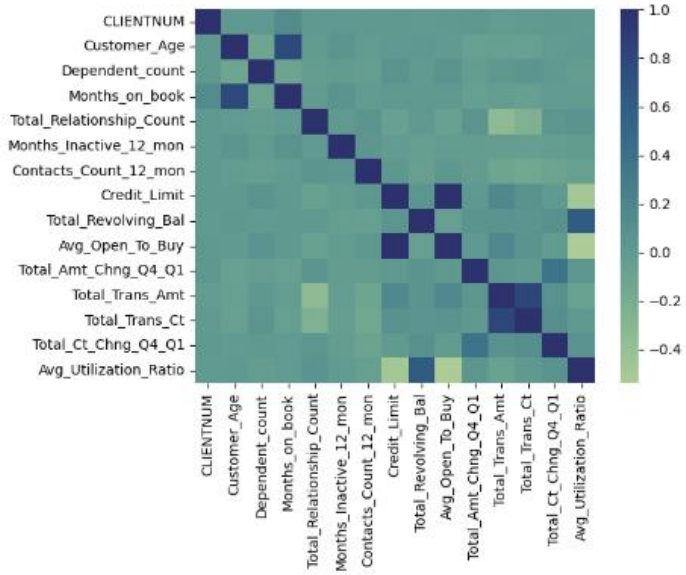
```
CLIENTNUM                 0
Attrition_Flag            0
Customer_Age              0
Gender                    0
Dependent_count           0
Education_Level           0
Marital_Status            0
Income_Category           0
Card_Category             0
Months_on_book            0
Total_Relationship_Count  0
Months_Inactive_12_mon    0
Contacts_Count_12_mon     0
Credit_Limit              0
Total_Revolving_Bal       0
Avg_Open_To_Buy           0
Total_Amt_Chng_Q4_Q1      0
Total_Trans_Amt           0
Total_Trans_Ct            0
Total_Ct_Chng_Q4_Q1       0
Avg_Utilization_Ratio     0
dtype: int64
```

Devorah Finer
Professor Rivillis
DS-399 Capstone
October 15th, 2024



[211]:
```
# Check for missing values
# Dummy variable for Customer Attrition
bank2 = bank.copy()

# Create dummy variable for Attrition_Flag
att = pd.get_dummies(bank2['Attrition_Flag'], drop_first=True)

# Drop original Attrition_Flag column
bank2 = bank2.drop('Attrition_Flag', axis=1)

# Merge the dummy variable into the original dataframe
bank2 = bank2.merge(att, how='left', left_index=True, right_index=True)

# Select only numeric columns for correlation
numeric_cols = bank2.select_dtypes(include=[float, int])

# Calculate correlation matrix
corr_matrix = numeric_cols.corr()

# Plot heatmap
sns.heatmap(corr_matrix, cmap='crest')
```

[211]: <Axes: >

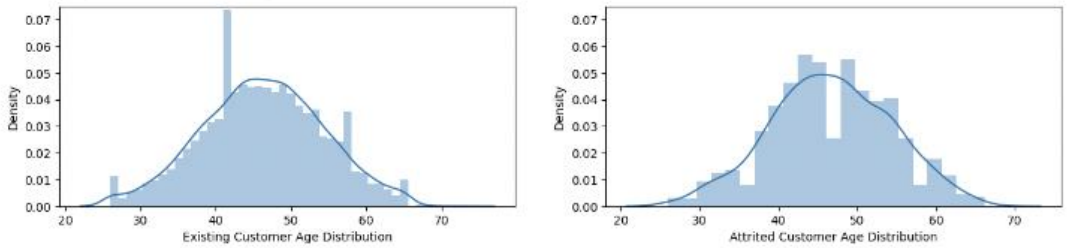


Devorah Finer
Professor Rivillis
DS-399 Capstone
October 15th, 2024

[215]:
```
stay = bank2[bank2['Existing Customer']== 1]
churn = bank2[bank2['Existing Customer'] == 0]

plt.figure(figsize=(15,3))
plt.subplot(1,2,1)
sns.distplot(stay.Customer_Age)
plt.ylim(0,0.075)
plt.xlabel('Existing Customer Age Distribution')

plt.subplot(1,2,2)
sns.distplot(churn.Customer_Age)
plt.ylim(0,0.075)
plt.xlabel('Attrited Customer Age Distribution')
```

```
/var/folders/2h/36l08yd119j08yg4pgvfrn900000gn/T/ipykernel_83129/4028311347.py:6: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(stay.Customer_Age)
/var/folders/2h/36l08yd119j08yg4pgvfrn900000gn/T/ipykernel_83129/4028311347.py:11: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(churn.Customer_Age)
```

[215]:
```
Text(0.5, 0, 'Attrited Customer Age Distribution')
```

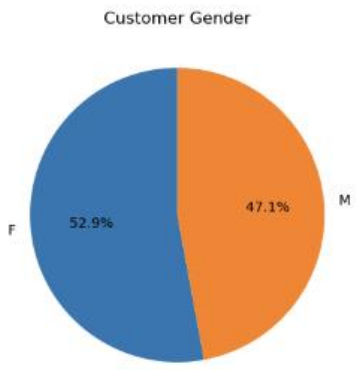


[217]:
```
existing_age = bank[bank.Attrition_Flag=='Existing Customer'].Customer_Age.sort_values()
np.percentile(existing_age, [20,50,80])

churn_age = bank[bank.Attrition_Flag=='Attrited Customer'].Customer_Age.sort_values()
np.percentile(churn_age, [20,50,80])
```

[217]:
```
array([40., 47., 54.])
```

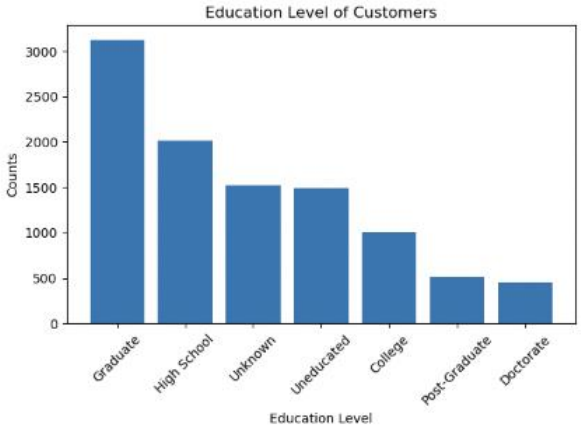
[219]:
```
gender = bank.Gender.value_counts().reset_index()
gender.columns=['Gender', 'Counts']

fig, ax = plt.subplots()
ax.pie(gender.Counts, labels = gender.Gender, autopct ='% .1f%%', startangle = 90)
ax.set_title('Customer Gender')
```

[219]:
```
Text(0.5, 1.0, 'Customer Gender')
```



Devorah Finer
Professor Rivillis
DS-399 Capstone
October 15th, 2024

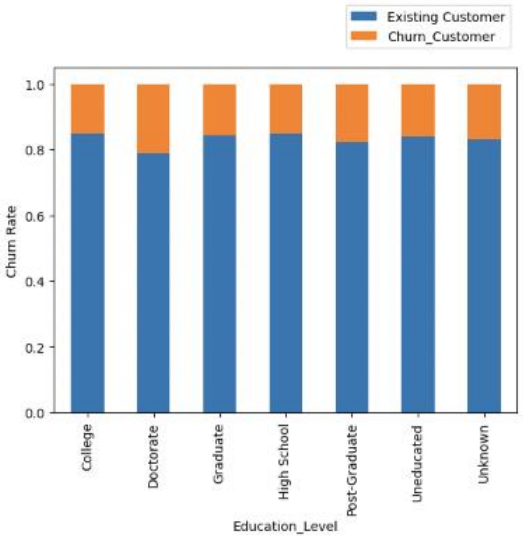
[225]:
```
ed_level = bank.Education_Level.value_counts().reset_index()
ed_level.columns = ["Education_Level", "Counts"]

fig, ax = plt.subplots()
ax.bar(ed_level["Education_Level"], ed_level["Counts"])
ax.set_ylabel('Counts')
ax.set_xlabel('Education Level')
ax.set_title('Education Level of Customers')

plt.xticks(rotation=45)  # Rotate the x labels for better readability
plt.tight_layout()       # Adjust the layout to ensure everything fits without overlapping
plt.show()
```



[227]:
```
education = bank2.pivot_table('Existing Customer', ['Education_Level']).reset_index()
education['Churn_Customer']= 1-education['Existing Customer']
print(education)

education.plot.bar(x='Education_Level', y=['Existing Customer', 'Churn_Customer'], stacked=True)
plt.legend(loc='upper right', bbox_to_anchor = (1, 1.2))
plt.ylabel("Churn Rate")
```

```
  Education_Level  Existing Customer  Churn_Customer
0         College           0.847976        0.152024
1       Doctorate           0.789357        0.210643
2        Graduate           0.844309        0.155691
3     High School           0.847988        0.152012
4   Post-Graduate           0.821705        0.178295
5      Uneducated           0.840619        0.159381
6         Unknown           0.831468        0.168532
```

[227]:
```
Text(0, 0.5, 'Churn Rate')
```

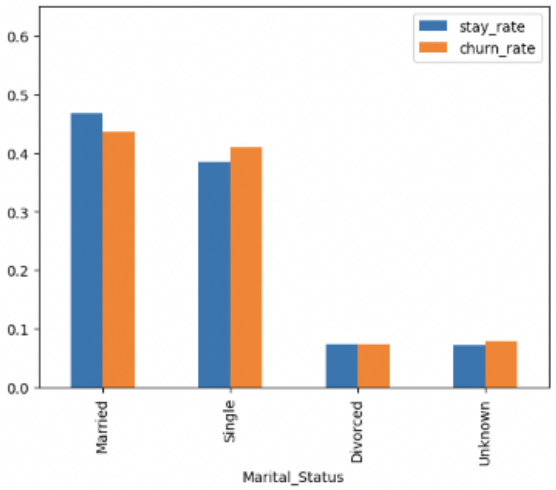


Devorah Finer
Professor Rivillis
DS-399 Capstone
October 15th, 2024

Marital Status

[235]:
```
stay_rship = stay.Marital_Status.value_counts()
churn_rship = churn.Marital_Status.value_counts()

rship = pd.concat([stay_rship, churn_rship], axis=1).reset_index()
rship.columns = ['Marital_Status','Stay', 'Churn']

rship['stay_rate']=rship.Stay/(rship.Stay.sum())
rship['churn_rate']=rship.Churn/(rship.Churn.sum())

rship = rship.drop(['Churn', 'Stay'], axis =1)
rship = rship.sort_values('stay_rate', ascending=False)


rship.plot(x='Marital_Status', kind='bar', stacked=False, ylim=(0,0.65))
```

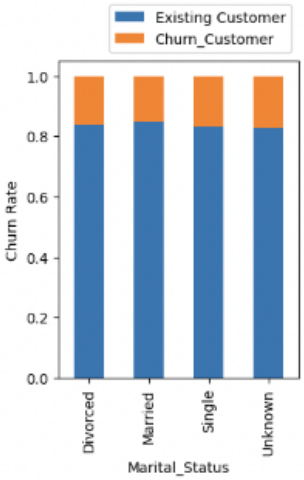
[235]:
```
<Axes: xlabel='Marital_Status'>
```



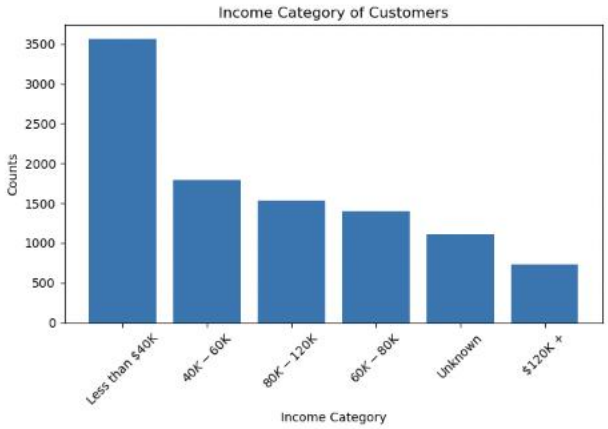
[237]:
```
rship = bank2.pivot_table('Existing Customer', ['Marital_Status']).reset_index()
rship['Churn_Customer']= 1-rship['Existing Customer']
print(rship)

rship.plot.bar(x='Marital_Status', y=['Existing Customer', 'Churn_Customer'], stacked=True, figsize=(3,4))
plt.legend(loc='upper right', bbox_to_anchor = (1, 1.2))
plt.ylabel("Churn Rate")
```

```
  Marital_Status  Existing Customer  Churn_Customer
0       Divorced           0.838235        0.161765
1        Married           0.848731        0.151269
2         Single           0.830586        0.169414
3        Unknown           0.827770        0.172230
```

[237]:
```
Text(0, 0.5, 'Churn Rate')
```

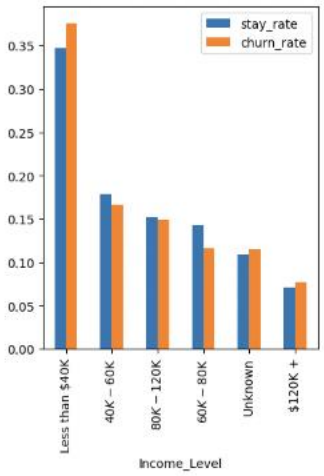


```
plt.show()
```



[243]:
```
stay_income = stay.Income_Category.value_counts()
churn_income = churn.Income_Category.value_counts()

income = pd.concat([stay_income, churn_income], axis=1).reset_index()
income.columns = ['Income_Level','Stay', 'Churn']

income['stay_rate']=income.Stay/(income.Stay.sum())
income['churn_rate']=income.Churn/(income.Churn.sum())

income = income.drop(['Churn', 'Stay'], axis =1)

income.plot(x='Income_Level', kind='bar', stacked=False, figsize=(4,5))
```

[243]:
```
<Axes: xlabel='Income_Level'>
```

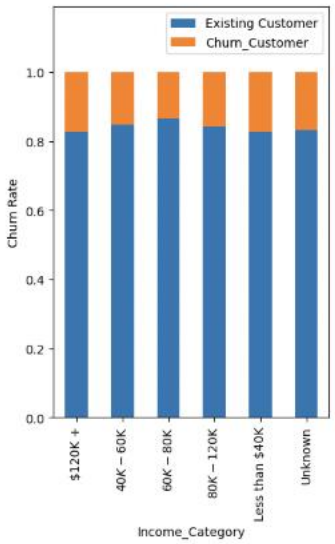


Devorah Finer
Professor Rivillis
DS-399 Capstone
October 15th, 2024

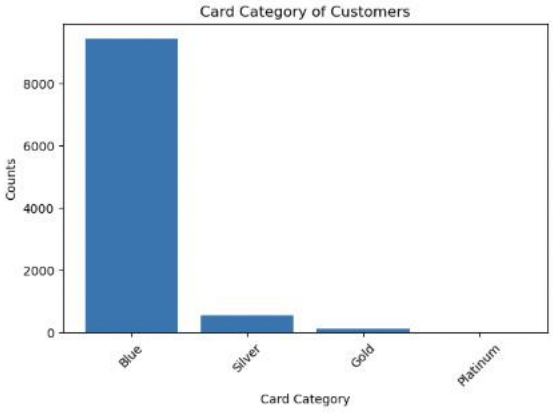
[245]:
```
income2 = bank2.pivot_table('Existing Customer', ['Income_Category']).reset_index()
income2['Churn_Customer']= 1-income2['Existing Customer']
print (income2)

income2.plot.bar(x='Income_Category', y=['Existing Customer', 'Churn_Customer'], stacked=True, figsize=(4,6), ylim=(0,1.19))
plt.ylabel("Churn Rate")
```

```
  Income_Category  Existing Customer  Churn_Customer
0         $120K +           0.826685        0.173315
1     $40K - $60K           0.848603        0.151397
2     $60K - $80K           0.865193        0.134807
3    $80K - $120K           0.842345        0.157655
4  Less than $40K           0.828138        0.171862
5         Unknown           0.831835        0.168165
```

[245]: `Text(0, 0.5, 'Churn Rate')`



[247]:
```
card_type = bank.Card_Category.value_counts().reset_index()
card_type.columns = ["Card_Category", "Counts"]

fig, ax = plt.subplots()
ax.bar(card_type["Card_Category"], card_type["Counts"])
ax.set_ylabel('Counts')
ax.set_xlabel('Card Category')
ax.set_title('Card Category of Customers')

plt.xticks(rotation=45)  # Rotate x labels for better readability
plt.tight_layout()       # Adjust the layout for a cleaner look
plt.show()
```

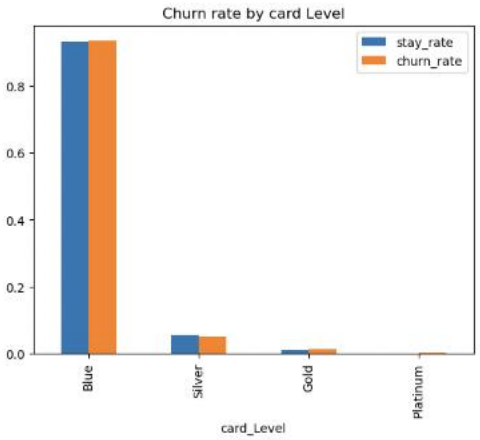


Devorah Finer
Professor Rivillis
DS-399 Capstone
October 15th, 2024

Card Category

[251]:
```
stay_card = stay.Card_Category.value_counts()
churn_card = churn.Card_Category.value_counts()

card = pd.concat([stay_card, churn_card], axis=1).reset_index()
card.columns = ['card_Level','Stay', 'Churn']

card['stay_rate']=card.Stay/(card.Stay.sum())
card['churn_rate']=card.Churn/(card.Churn.sum())

card = card.drop(['Churn', 'Stay'], axis =1)

card.plot(x='card_Level', kind='bar', stacked=False, title='Churn rate by card Level')
```

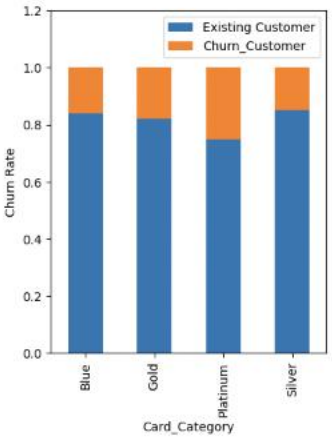
[251]:
```
<Axes: title={'center': 'Churn rate by card Level'}, xlabel='card_Level'>
```



[253]:
```
card = bank2.pivot_table('Existing Customer', ['Card_Category']).reset_index()
card['Churn_Customer']= 1-card['Existing Customer']
print (card)

card.plot.bar(x='Card_Category', y=['Existing Customer', 'Churn_Customer'], stacked=True, figsize= (4,5), ylim=(0,1.2))
plt.ylabel("Churn Rate")
```

```
  Card_Category  Existing Customer  Churn_Customer
0          Blue           0.839021        0.160979
1          Gold           0.818966        0.181034
2      Platinum           0.750000        0.250000
3        Silver           0.852252        0.147748
```

[253]:
```
Text(0, 0.5, 'Churn Rate')
```

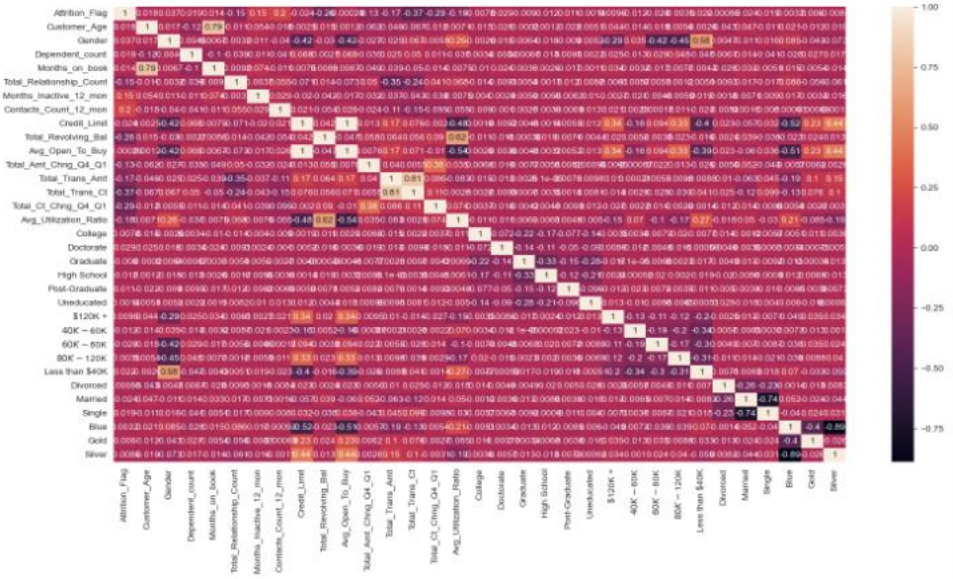


Devorah Finer
Professor Rivillis
DS-399 Capstone
October 15th, 2024

[186]:
```
# One-hot encoding for categorical columns
c_data.Attrition_Flag = c_data.Attrition_Flag.replace({'Attrited Customer': 1, 'Existing Customer': 0})
c_data.Gender = c_data.Gender.replace({'F': 1, 'M': 0})
c_data = pd.concat([c_data, pd.get_dummies(c_data['Education_Level']).drop(columns=['Unknown'])], axis=1)
c_data = pd.concat([c_data, pd.get_dummies(c_data['Income_Category']).drop(columns=['Unknown'])], axis=1)
c_data = pd.concat([c_data, pd.get_dummies(c_data['Marital_Status']).drop(columns=['Unknown'])], axis=1)
c_data = pd.concat([c_data, pd.get_dummies(c_data['Card_Category']).drop(columns=['Platinum'])], axis=1)
c_data.drop(columns=['Education_Level', 'Income_Category', 'Marital_Status', 'Card_Category', 'CLIENTNUM'], inplace=True)
```

```
/var/folders/2h/36l08yd119j08yg4pgvfrn900000gn/T/ipykernel_68293/435428284.py:2: FutureWarning:

Downcasting behavior in `replace` is deprecated and will be removed in a future version. To retain the old behavior, explicitly call `re
sult.infer_objects(copy=False)`. To opt-in to the future behavior, set `pd.set_option('future.no_silent_downcasting', True)`

/var/folders/2h/36l08yd119j08yg4pgvfrn900000gn/T/ipykernel_68293/435428284.py:3: FutureWarning:

Downcasting behavior in `replace` is deprecated and will be removed in a future version. To retain the old behavior, explicitly call `re
sult.infer_objects(copy=False)`. To opt-in to the future behavior, set `pd.set_option('future.no_silent_downcasting', True)`
```

[188]:
```
# Correlation heatmap of processed data
sns.heatmap(c_data.corr('pearson'), annot=True)
plt.show()
```



[190]:
```
# Balance the data using SMOTE
oversample = SMOTE()
X, y = oversample.fit_resample(c_data[c_data.columns[1:]], c_data[c_data.columns[0]])
usampled_df = X.assign(Churn=y)
```

[192]:
```
# Apply PCA to reduce dimensions
N_COMPONENTS = 4
pca_model = PCA(n_components=N_COMPONENTS)
pc_matrix = pca_model.fit_transform(usampled_df)
usampled_df_with_pcs = pd.concat([usampled_df, pd.DataFrame(pc_matrix, columns=[f'PC-{i}' for i in range(N_COMPONENTS)])], axis=1)
```

[194]:
```
# Display transformed dataset with principal components
usampled_df_with_pcs.head()
```

[194]:

	Customer_Age	Gender	Dependent_count	Months_on_book	Total_Relationship_Count	Months_Inactive_12_mon	Contacts_Count_12_mon	Credit_Limit
0	45	0	3	39	5	1	3	12691.0
1	49	1	5	44	6	1	2	8256.0
2	51	0	3	36	4	1	0	3418.0
3	40	1	4	34	3	4	1	3313.0
4	40	0	3	21	5	1	0	4716.0

5 rows × 37 columns

[196]:
```
# Split data into features and target
X_features = ['Total_Trans_Ct', 'PC-3', 'PC-1', 'PC-0', 'PC-2', 'Total_Ct_Chng_Q4_Q1', 'Total_Relationship_Count']
X = usampled_df_with_pcs[X_features]
y = usampled_df_with_pcs['Churn']
```

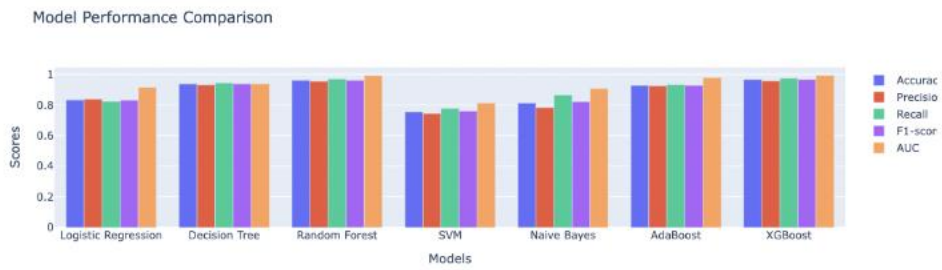
Devorah Finer
Professor Rivillis
DS-399 Capstone
October 15th, 2024

[165]:
```
# Install necessary libraries
!pip install xgboost

# Import the necessary modules
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
import xgboost as xgb
import pandas as pd

# Define your models, including XGBoost
models = {
    'Logistic Regression': LogisticRegression(),
    'Decision Tree': DecisionTreeClassifier(),
    'Random Forest': RandomForestClassifier(),
    'SVM': SVC(probability=True),
    'Naive Bayes': GaussianNB(),
    'AdaBoost': AdaBoostClassifier(),
    'XGBoost': xgb.XGBClassifier(use_label_encoder=False, eval_metric='logloss')
}

# Split data (assuming X and y have been defined earlier)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Dictionary to store metrics
metrics = {
    'Model': [],
    'Accuracy': [],
    'Precision': [],
    'Recall': [],
    'F1-score': [],
    'AUC': []
}

# Iterate through each model
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    y_prob = model.predict_proba(X_test)[:, 1] if hasattr(model, 'predict_proba') else None

    # Calculate metrics
    acc = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    auc = roc_auc_score(y_test, y_prob) if y_prob is not None else 'N/A'

    # Append the results
    metrics['Model'].append(name)
    metrics['Accuracy'].append(acc)
    metrics['Precision'].append(precision)
    metrics['Recall'].append(recall)
    metrics['F1-score'].append(f1)
    metrics['AUC'].append(auc)

# Convert metrics to a DataFrame
metrics_df = pd.DataFrame(metrics)
print(metrics_df)
```

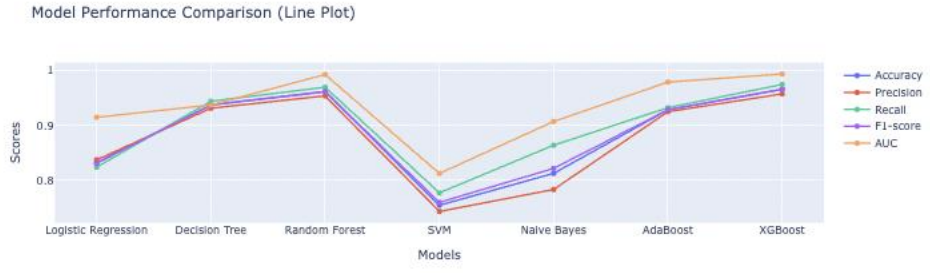
```
Requirement already satisfied: xgboost in /opt/anaconda3/lib/python3.12/site-packages (2.1.1)
Requirement already satisfied: numpy in /opt/anaconda3/lib/python3.12/site-packages (from xgboost) (1.26.4)
Requirement already satisfied: scipy in /opt/anaconda3/lib/python3.12/site-packages (from xgboost) (1.13.1)
/opt/anaconda3/lib/python3.12/site-packages/sklearn/linear_model/_logistic.py:469: 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

/opt/anaconda3/lib/python3.12/site-packages/sklearn/ensemble/_weight_boosting.py:519: FutureWarning:

The SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent this warning.

                Model  Accuracy  Precision    Recall  F1-score       AUC
0  Logistic Regression  0.832353   0.837655  0.824176  0.830861  0.914977
1        Decision Tree  0.937451   0.931475  0.944270  0.937829  0.937456
2        Random Forest  0.961569   0.954054  0.969780  0.961853  0.992925
3                  SVM  0.754902   0.743619  0.777473  0.760169  0.812803
4          Naive Bayes  0.812941   0.783630  0.864207  0.821948  0.907394
5             AdaBoost  0.928627   0.925234  0.932496  0.928851  0.979161
6              XGBoost  0.965882   0.957594  0.974882  0.966161  0.993844
/opt/anaconda3/lib/python3.12/site-packages/xgboost/core.py:158: UserWarning:

[18:45:14] WARNING: /Users/runner/work/xgboost/xgboost/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.
```

Devorah Finer
Professor Rivillis
DS-399 Capstone
October 15th, 2024

[166]:
```
import plotly.graph_objs as go

# Prepare traces for each metric
accuracy_trace = go.Bar(x=metrics_df['Model'], y=metrics_df['Accuracy'], name='Accuracy')
precision_trace = go.Bar(x=metrics_df['Model'], y=metrics_df['Precision'], name='Precision')
recall_trace = go.Bar(x=metrics_df['Model'], y=metrics_df['Recall'], name='Recall')
f1_trace = go.Bar(x=metrics_df['Model'], y=metrics_df['F1-score'], name='F1-score')
auc_trace = go.Bar(x=metrics_df['Model'], y=metrics_df['AUC'], name='AUC')

# Create layout
layout = go.Layout(
    title='Model Performance Comparison',
    xaxis=dict(title='Models'),
    yaxis=dict(title='Scores'),
    barmode='group'
)

# Create figure
fig = go.Figure(data=[accuracy_trace, precision_trace, recall_trace, f1_trace, auc_trace], layout=layout)

# Show plot
fig.show()
```



[167]:
```
import plotly.graph_objs as go

# Prepare traces for each metric as lines
accuracy_trace = go.Scatter(x=metrics_df['Model'], y=metrics_df['Accuracy'], mode='lines+markers', name='Accuracy')
precision_trace = go.Scatter(x=metrics_df['Model'], y=metrics_df['Precision'], mode='lines+markers', name='Precision')
recall_trace = go.Scatter(x=metrics_df['Model'], y=metrics_df['Recall'], mode='lines+markers', name='Recall')
f1_trace = go.Scatter(x=metrics_df['Model'], y=metrics_df['F1-score'], mode='lines+markers', name='F1-score')
auc_trace = go.Scatter(x=metrics_df['Model'], y=metrics_df['AUC'], mode='lines+markers', name='AUC')

# Create layout
layout = go.Layout(
    title='Model Performance Comparison (Line Plot)',
    xaxis=dict(title='Models'),
    yaxis=dict(title='Scores'),
    hovermode='closest'
)

# Create figure
fig = go.Figure(data=[accuracy_trace, precision_trace, recall_trace, f1_trace, auc_trace], layout=layout)

# Show plot
fig.show()
```



Devorah Finer
Professor Rivillis
DS-399 Capstone
October 15th, 2024

[206]:
```
# Importing necessary libraries for XGBoost
!pip install xgboost
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, roc_auc_score, f1_score, classification_report

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    usampled_df_with_pcs.drop(columns=['Churn']),
    usampled_df_with_pcs['Churn'],
    test_size=0.2,
    random_state=42
)
```

```
Requirement already satisfied: xgboost in /opt/anaconda3/lib/python3.12/site-packages (2.1.1)
Requirement already satisfied: numpy in /opt/anaconda3/lib/python3.12/site-packages (from xgboost) (1.26.4)
Requirement already satisfied: scipy in /opt/anaconda3/lib/python3.12/site-packages (from xgboost) (1.13.1)
```

[226]:
```
# Standardize the features for XGBoost
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

[208]:
```
# XGBoost model
xgb_model = XGBClassifier(n_estimators=100, max_depth=4, learning_rate=0.1, random_state=42)
xgb_model.fit(X_train_scaled, y_train)
```

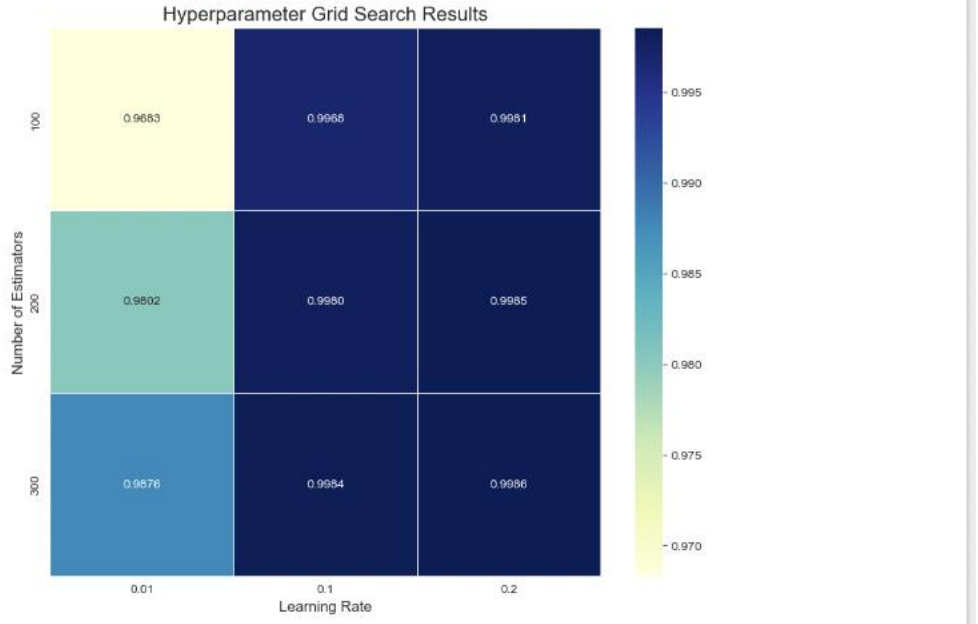
[208]:
```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.1, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=4, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=100, n_jobs=None,
              num_parallel_tree=None, random_state=42, ...)
```

Devorah Finer
Professor Rivillis
DS-399 Capstone
October 15th, 2024

[256]:
```
from sklearn.model_selection import GridSearchCV

# Define the parameter grid
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [3, 4, 5],
    'learning_rate': [0.01, 0.1, 0.2],
    'subsample': [0.8, 1.0]
}

# Initialize the XGBoost model
xgb_model_tune = XGBClassifier(random_state=42)

# Grid search for hyperparameter tuning
grid_search = GridSearchCV(estimator=xgb_model_tune, param_grid=param_grid, cv=3, scoring='roc_auc', verbose=2, n_jobs=-1)
grid_search.fit(X_train_scaled, y_train)

# Best parameters from Grid Search
print(f"Best parameters found: {grid_search.best_params_}")
```

```
Fitting 3 folds for each of 54 candidates, totalling 162 fits
Best parameters found: {'learning_rate': 0.2, 'max_depth': 4, 'n_estimators': 300, 'subsample': 1.0}
```

[257]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# Assuming 'grid_search' is your GridSearchCV object
# Convert the GridSearchCV results into a DataFrame
results = pd.DataFrame(grid_search.cv_results_)

# Create a pivot table to structure data for the heatmap
heatmap_data = results.pivot_table(
    index='param_n_estimators',
    columns='param_learning_rate',
    values='mean_test_score'
)

# Plotting the heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(heatmap_data, annot=True, cmap='YlGnBu', fmt=".4f", linewidths=0.5)
plt.title('Hyperparameter Grid Search Results', fontsize=16)
plt.xlabel('Learning Rate', fontsize=12)
plt.ylabel('Number of Estimators', fontsize=12)
plt.show()
```



Devorah Finer
Professor Rivillis
DS-399 Capstone
October 15th, 2024

[245]:
```
# XGBoost model with tuned hyperparameters
xgb_model = XGBClassifier(
    n_estimators=300,
    max_depth=4,
    learning_rate=0.2,
    subsample=1.0,
    random_state=42
)

# Fit the model
xgb_model.fit(X_train_scaled, y_train)
```

[245]:
```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.2, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=4, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=300, n_jobs=None,
              num_parallel_tree=None, random_state=42, ...)
```

[248]:
```
# Predictions
y_pred = xgb_model.predict(X_test_scaled)
y_pred_proba = xgb_model.predict_proba(X_test_scaled)[:, 1]
```

[250]:
```
# Evaluation
accuracy = accuracy_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
auc = roc_auc_score(y_test, y_pred_proba)
```

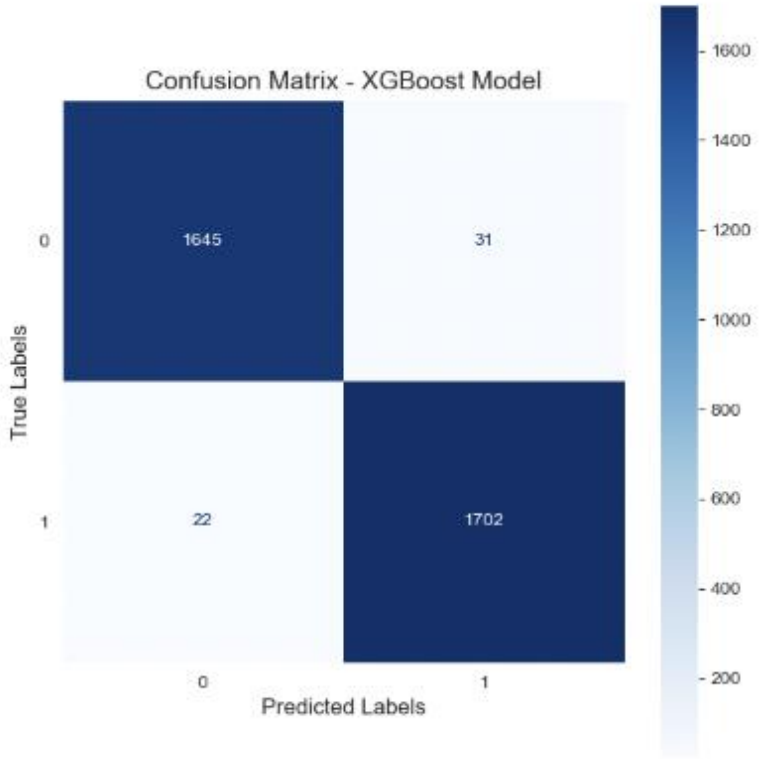
[252]:
```
print(f"Accuracy: {accuracy:.4f}")
print(f"F1 Score: {f1:.4f}")
print(f"AUC: {auc:.4f}")
```

```
Accuracy: 0.9844
F1 Score: 0.9847
AUC: 0.9989
```

[254]:
```
# Classification report
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

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

           0       0.99      0.98      0.98      1676
           1       0.98      0.99      0.98      1724

    accuracy                           0.98      3400
   macro avg       0.98      0.98      0.98      3400
weighted avg       0.98      0.98      0.98      3400
```

62]:
```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

# Confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Plotting the confusion matrix with improvements
fig, ax = plt.subplots(figsize=(6, 6))  # Adjust figure size for clarity
disp = ConfusionMatrixDisplay(confusion_matrix=cm)

# Improved display of the confusion matrix
disp.plot(cmap='Blues', ax=ax, values_format='d')

# Adding more details to the plot for clarity
plt.title('Confusion Matrix - XGBoost Model', fontsize=14)
plt.xlabel('Predicted Labels', fontsize=12)
plt.ylabel('True Labels', fontsize=12)
plt.grid(False)  # Remove grid lines for better visibility
plt.tight_layout()  # Adjust layout for better spacing
plt.show()
```



Devorah Finer
Professor Rivillis
DS-399 Capstone
October 15th, 2024

[264]:
```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

# Get the predicted probabilities for the positive class (1)
y_proba = xgb_model.predict_proba(X_test_scaled)[:, 1]

# Calculate the false positive rate (fpr), true positive rate (tpr), and thresholds
fpr, tpr, thresholds = roc_curve(y_test, y_proba)

# Calculate the AUC (Area Under the Curve)
roc_auc = auc(fpr, tpr)

# Plotting the ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC curve (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='gray', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```