

Bank Data Churn

February 28, 2024

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

# Load the dataset
df = pd.read_csv(r'C:\Users\jerme\Downloads\Data Science_
↳project\churn_prediction.csv')

# Display the first few rows
print(df.head())

# Summary statistics
print(df.describe())

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

# Visualize the distribution of the 'age' feature
sns.histplot(df['age'], kde=True)
plt.title('Age Distribution')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()

# Visualize the distribution of the 'current_balance' feature
sns.histplot(df['current_balance'], kde=True)
plt.title('Current Balance Distribution')
plt.xlabel('Current Balance')
plt.ylabel('Frequency')
plt.show()
```

	customer_id	vintage	age	gender	dependents	occupation	city	\
0	1	3135	66	Male	0.0	self_employed	187.0	
1	2	310	35	Male	0.0	self_employed	NaN	
2	4	2356	31	Male	0.0	salaried	146.0	
3	5	478	90	NaN	NaN	self_employed	1020.0	
4	6	2531	42	Male	2.0	self_employed	1494.0	

	customer_nw_category	branch_code	days_since_last_transaction	...	\
0	2	755		224.0	...
1	2	3214		60.0	...
2	2	41		NaN	...
3	2	582		147.0	...
4	3	388		58.0	...

	previous_month_end_balance	average_monthly_balance_prevQ	\
0	1458.71	1458.71	
1	8704.66	7799.26	
2	5815.29	4910.17	
3	2291.91	2084.54	
4	1401.72	1643.31	

	average_monthly_balance_prevQ2	current_month_credit	\
0	1449.07	0.20	
1	12419.41	0.56	
2	2815.94	0.61	
3	1006.54	0.47	
4	1871.12	0.33	

	previous_month_credit	current_month_debit	previous_month_debit	\
0	0.20	0.20	0.20	
1	0.56	5486.27	100.56	
2	0.61	6046.73	259.23	
3	0.47	0.47	2143.33	
4	714.61	588.62	1538.06	

	current_month_balance	previous_month_balance	churn
0	1458.71	1458.71	0
1	6496.78	8787.61	0
2	5006.28	5070.14	0
3	2291.91	1669.79	1
4	1157.15	1677.16	1

[5 rows x 21 columns]

	customer_id	vintage	age	dependents	city	\
count	28382.000000	28382.000000	28382.000000	25919.000000	27579.000000	
mean	15143.508667	2364.336446	48.208336	0.347236	796.109576	
std	8746.454456	1610.124506	17.807163	0.997661	432.872102	
min	1.000000	180.000000	1.000000	0.000000	0.000000	
25%	7557.250000	1121.000000	36.000000	0.000000	409.000000	
50%	15150.500000	2018.000000	46.000000	0.000000	834.000000	
75%	22706.750000	3176.000000	60.000000	0.000000	1096.000000	
max	30301.000000	12899.000000	90.000000	52.000000	1649.000000	

	customer_nw_category	branch_code	days_since_last_transaction	\
count	28382.000000	28382.000000		25159.000000

mean	2.225530	925.975019	69.997814
std	0.660443	937.799129	86.341098
min	1.000000	1.000000	0.000000
25%	2.000000	176.000000	11.000000
50%	2.000000	572.000000	30.000000
75%	3.000000	1440.000000	95.000000
max	3.000000	4782.000000	365.000000

	current_balance	previous_month_end_balance \
count	2.838200e+04	2.838200e+04
mean	7.380552e+03	7.495771e+03
std	4.259871e+04	4.252935e+04
min	-5.503960e+03	-3.149570e+03
25%	1.784470e+03	1.906000e+03
50%	3.281255e+03	3.379915e+03
75%	6.635820e+03	6.656535e+03
max	5.905904e+06	5.740439e+06

	average_monthly_balance_prevQ	average_monthly_balance_prevQ2 \
count	2.838200e+04	2.838200e+04
mean	7.496780e+03	7.124209e+03
std	4.172622e+04	4.457581e+04
min	1.428690e+03	-1.650610e+04
25%	2.180945e+03	1.832507e+03
50%	3.542865e+03	3.359600e+03
75%	6.666887e+03	6.517960e+03
max	5.700290e+06	5.010170e+06

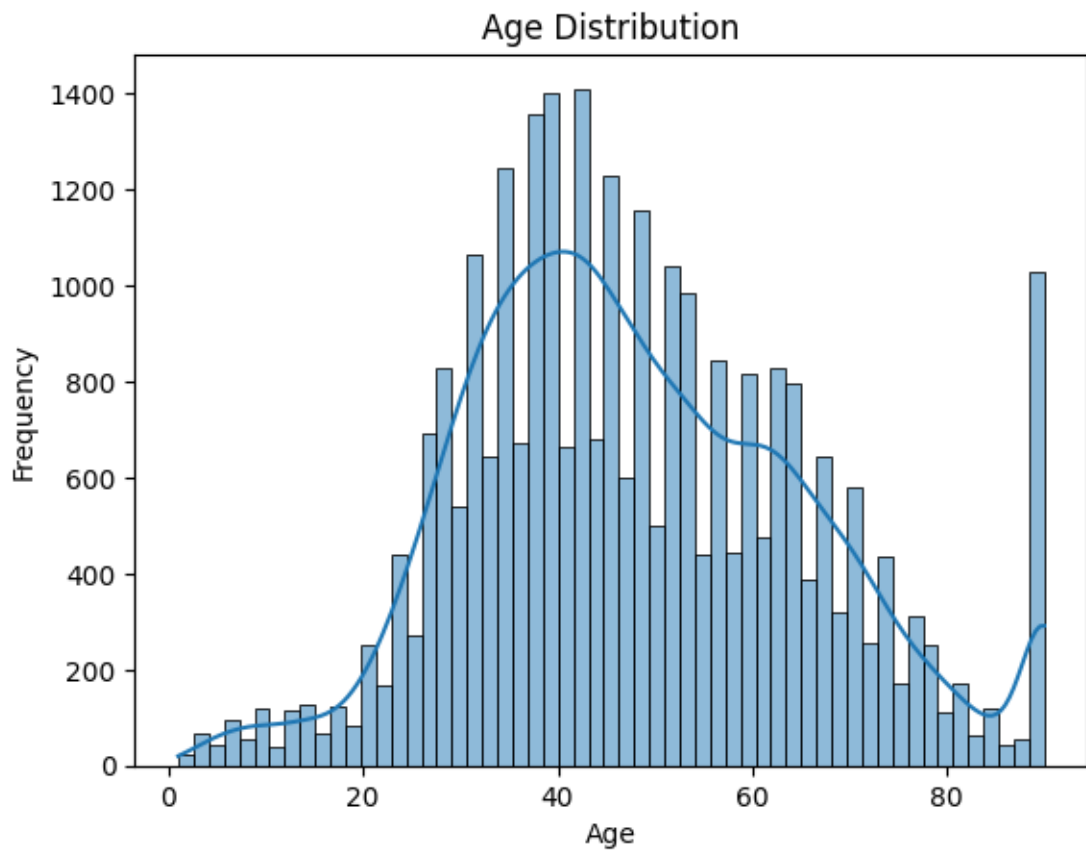
	current_month_credit	previous_month_credit	current_month_debit \
count	2.838200e+04	2.838200e+04	2.838200e+04
mean	3.433252e+03	3.261694e+03	3.658745e+03
std	7.707145e+04	2.968889e+04	5.198542e+04
min	1.000000e-02	1.000000e-02	1.000000e-02
25%	3.100000e-01	3.300000e-01	4.100000e-01
50%	6.100000e-01	6.300000e-01	9.193000e+01
75%	7.072725e+02	7.492350e+02	1.360435e+03
max	1.226985e+07	2.361808e+06	7.637857e+06

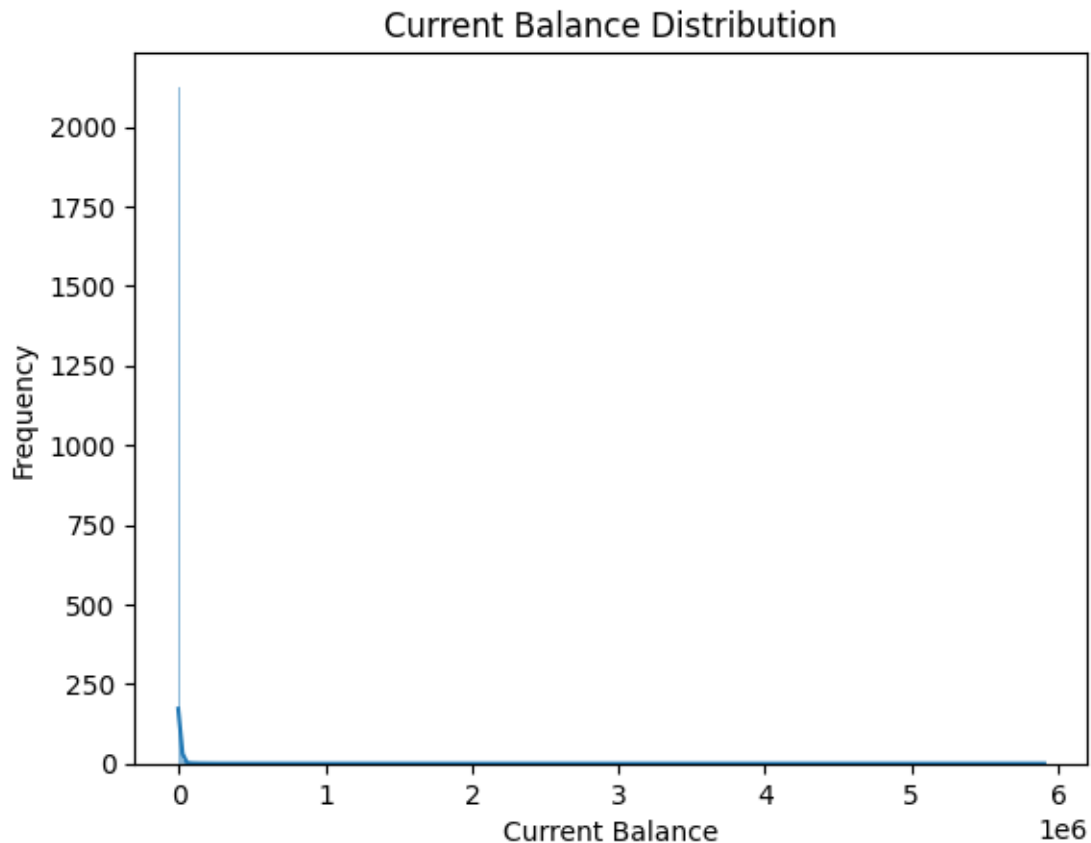
	previous_month_debit	current_month_balance	previous_month_balance \
count	2.838200e+04	2.838200e+04	2.838200e+04
mean	3.339761e+03	7.451133e+03	7.495177e+03
std	2.430111e+04	4.203394e+04	4.243198e+04
min	1.000000e-02	-3.374180e+03	-5.171920e+03
25%	4.100000e-01	1.996765e+03	2.074407e+03
50%	1.099600e+02	3.447995e+03	3.465235e+03
75%	1.357553e+03	6.667958e+03	6.654693e+03
max	1.414168e+06	5.778185e+06	5.720144e+06

```

                churn
count  28382.000000
mean    0.185329
std     0.388571
min     0.000000
25%     0.000000
50%     0.000000
75%     0.000000
max     1.000000
customer_id      0
vintage          0
age             0
gender          525
dependents      2463
occupation       80
city            803
customer_nw_category  0
branch_code      0
days_since_last_transaction  3223
current_balance  0
previous_month_end_balance  0
average_monthly_balance_prevQ  0
average_monthly_balance_prevQ2  0
current_month_credit  0
previous_month_credit  0
current_month_debit  0
previous_month_debit  0
current_month_balance  0
previous_month_balance  0
churn            0
dtype: int64

```





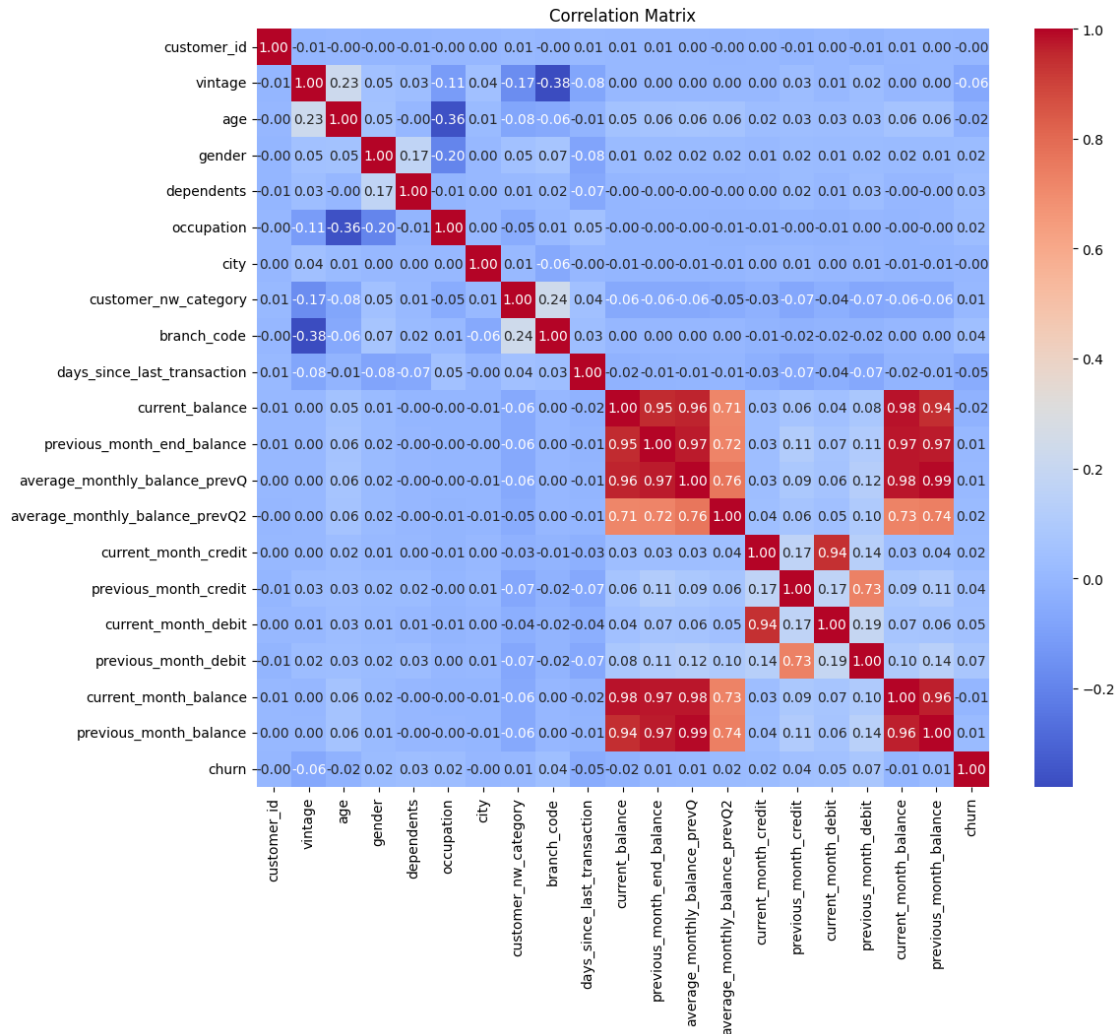
```
[2]: from sklearn.preprocessing import StandardScaler, LabelEncoder

# Fill missing values
df.fillna(method='ffill', inplace=True)

# Encode categorical variables
le = LabelEncoder()
df['gender'] = le.fit_transform(df['gender'].astype(str))
df['occupation'] = le.fit_transform(df['occupation'].astype(str))

# Normalize numerical features
scaler = StandardScaler()
numerical_features = ['vintage', 'age', 'dependents', 'customer_nw_category',
    ↳ 'days_since_last_transaction', 'current_balance',
    ↳ 'previous_month_end_balance', 'average_monthly_balance_prevQ',
    ↳ 'average_monthly_balance_prevQ2', 'current_month_credit',
    ↳ 'previous_month_credit', 'current_month_debit', 'previous_month_debit',
    ↳ 'current_month_balance', 'previous_month_balance']
df[numerical_features] = scaler.fit_transform(df[numerical_features])
```

```
# Visualize the correlation matrix
plt.figure(figsize=(12, 10))
sns.heatmap(df.corr(), annot=True, fmt='.2f', cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```



```
[3]: from sklearn.model_selection import train_test_split

X = df.drop(['churn', 'customer_id'], axis=1)
y = df['churn']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    ↪random_state=42)
```

```
[11]: from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

# Initialize and train the classifier
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Predictions
y_pred = model.predict(X_test)

# Evaluate the model
print(f'Accuracy: {accuracy_score(y_test, y_pred)}')
print(classification_report(y_test, y_pred))
```

Accuracy: 0.8682402677470495

	precision	recall	f1-score	support
0	0.88	0.97	0.92	4639
1	0.74	0.43	0.55	1038
accuracy			0.87	5677
macro avg	0.81	0.70	0.73	5677
weighted avg	0.86	0.87	0.85	5677

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

# Define the parameter grid
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [5, 10, 15],
    'min_samples_split': [2, 5, 10]
}

# Initialize the grid search model
grid_search = GridSearchCV(estimator=RandomForestClassifier(random_state=42),
    ↪param_grid=param_grid, cv=3, n_jobs=-1, verbose=2)

# Fit the grid search to the data
grid_search.fit(X_train, y_train)

# Print the best parameters and best score
print(f"Best Parameters: {grid_search.best_params_}")
print(f"Best Score: {grid_search.best_score_}")
```

Fitting 3 folds for each of 27 candidates, totalling 81 fits

Best Parameters: {'max_depth': 15, 'min_samples_split': 10, 'n_estimators': 100}

Best Score: 0.8643471022663843


```
[9]: import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
# Load the dataset
df = pd.read_csv(r'C:\Users\jerme\Downloads\Data Science_
↳project\churn_prediction.csv')
# Visualize the distribution of churn
plt.figure(figsize=(7, 4))
sns.countplot(x='churn', data=df)
plt.title('Distribution of Churn')
plt.xlabel('Churn')
plt.ylabel('Count')
plt.show()
# Visualize the distribution of age for churned and retained customers
plt.figure(figsize=(14, 7))
plt.subplot(1, 2, 1)
sns.histplot(df[df['churn'] == 0]['age'], kde=True, color='green', label='Not_
↳Churned')
plt.title('Age Distribution for Not Churned Customers')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.legend()

plt.subplot(1, 2, 2)
sns.histplot(df[df['churn'] == 1]['age'], kde=True, color='red', label='Churned')
plt.title('Age Distribution for Churned Customers')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.legend()
plt.show()

# Visualize the relationship between balance and churn
plt.figure(figsize=(7, 4))
sns.boxplot(x='churn', y='current_balance', data=df)
plt.title('Current Balance vs Churn')
plt.xlabel('Churn')
plt.ylabel('Current Balance')
plt.show()

# Feature importance from the model
feature_importances = pd.Series(model.feature_importances_, index=X.columns)
plt.figure(figsize=(12, 6))
feature_importances.nlargest(10).sort_values().plot(kind='barh')
plt.title('Top 10 Feature Importances')
plt.xlabel('Feature Importance Score')
plt.ylabel('Features')
plt.show()
```

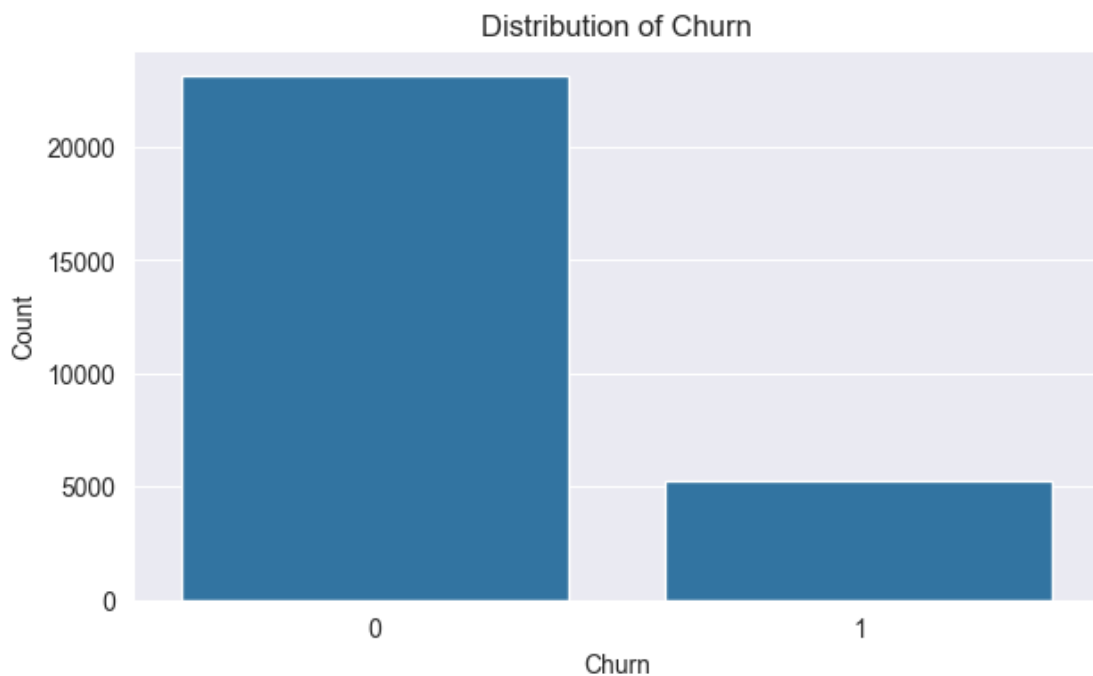
```

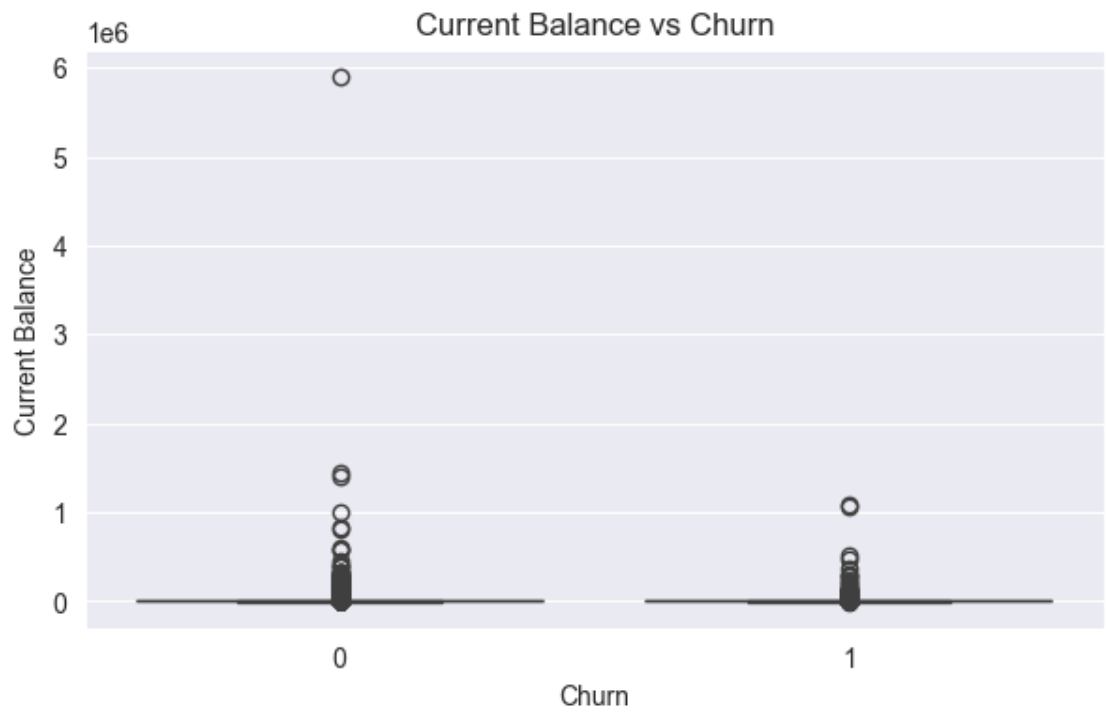
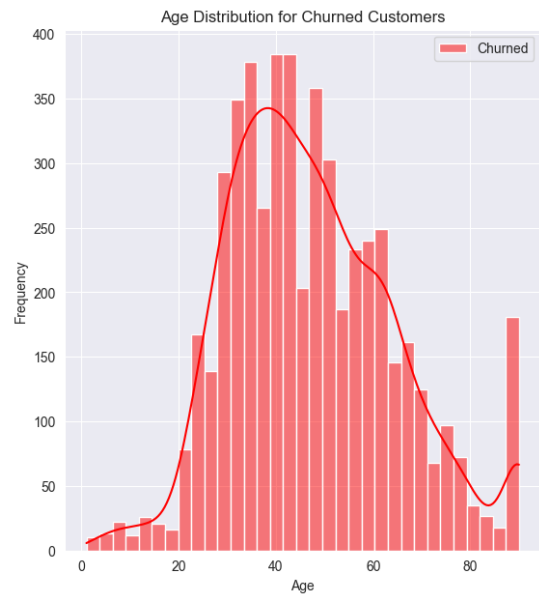
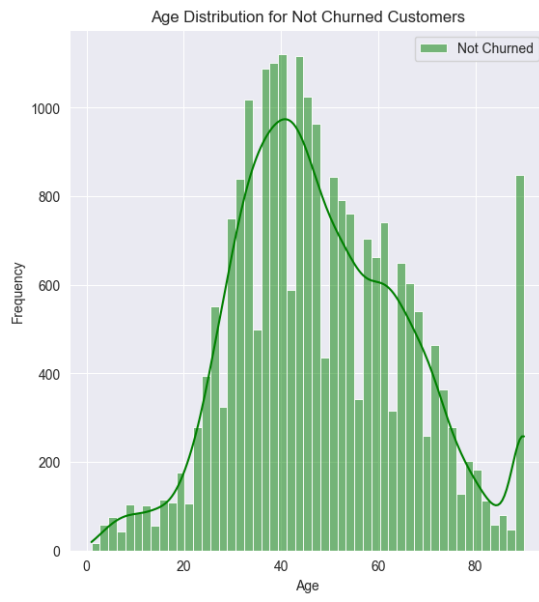
# ROC Curve
from sklearn.metrics import roc_curve, auc

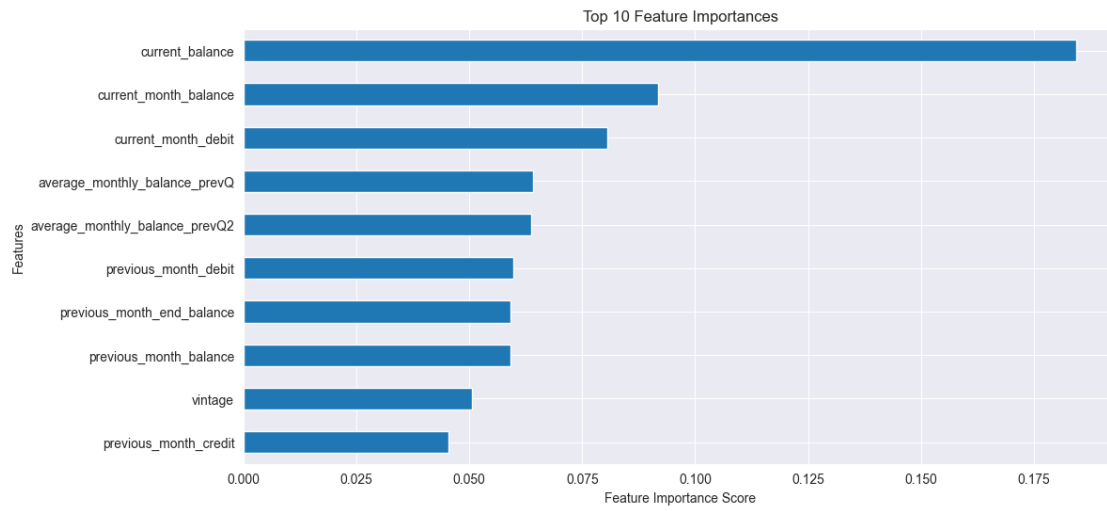
# Calculate the fpr and tpr for all thresholds of the classification
probs = model.predict_proba(X_test)
preds = probs[:, 1]
fpr, tpr, threshold = roc_curve(y_test, preds)
roc_auc = auc(fpr, tpr)

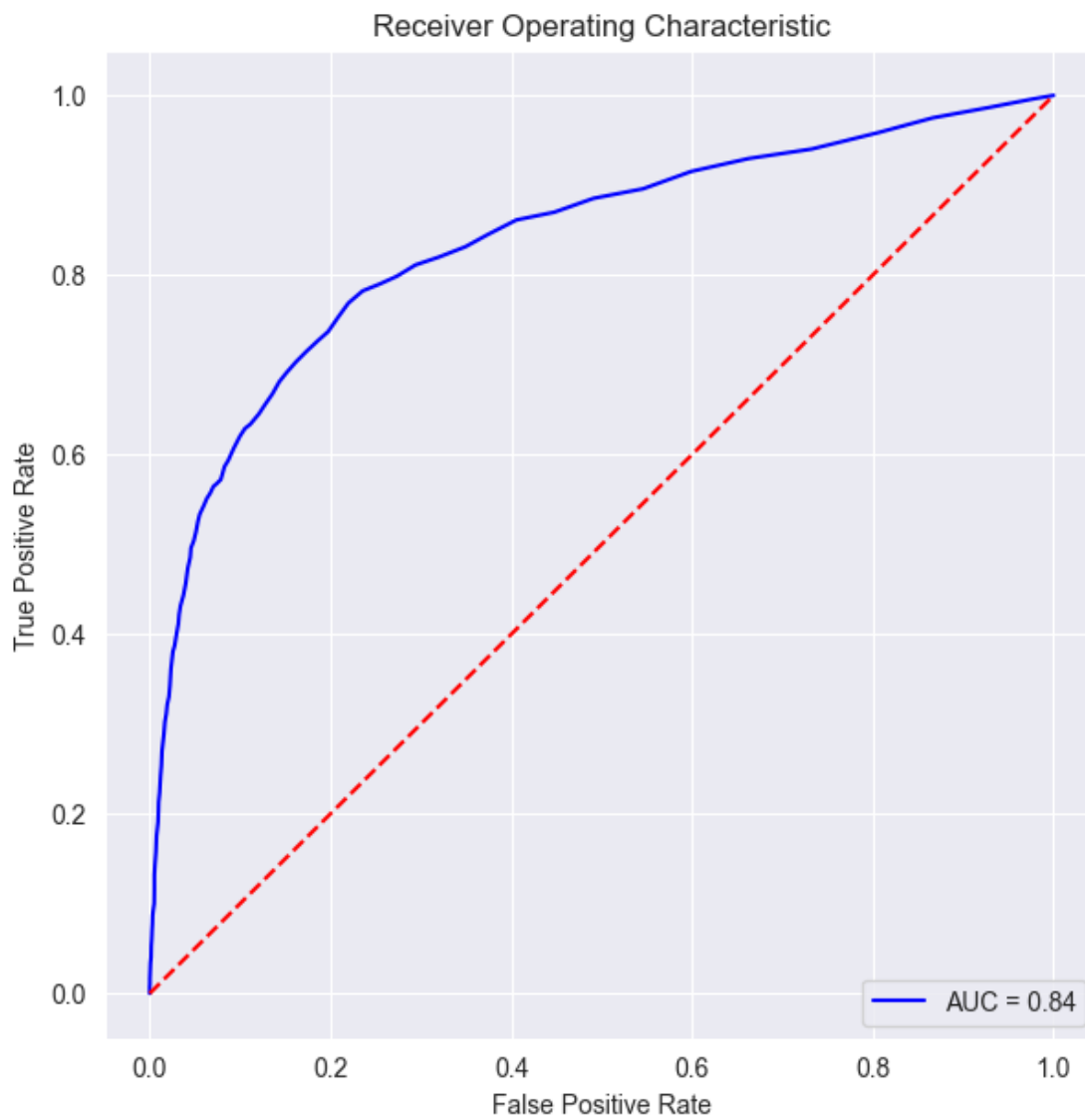
plt.figure(figsize=(7, 7))
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.plot([0, 1], [0, 1], 'r--')
plt.title('Receiver Operating Characteristic')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc = 'lower right')
plt.show()

```





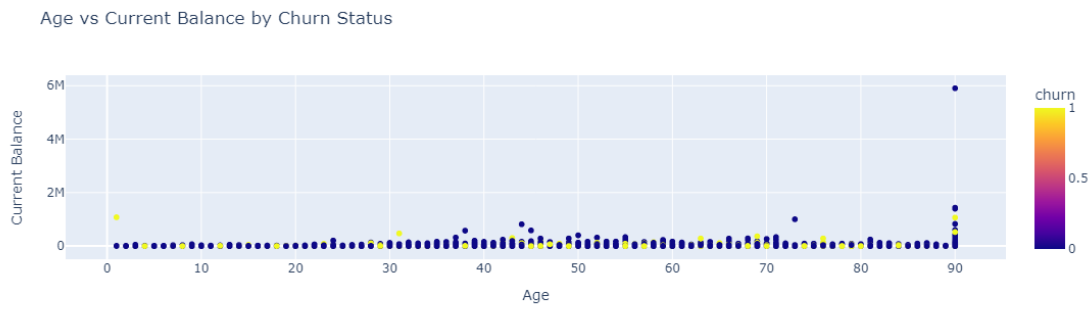




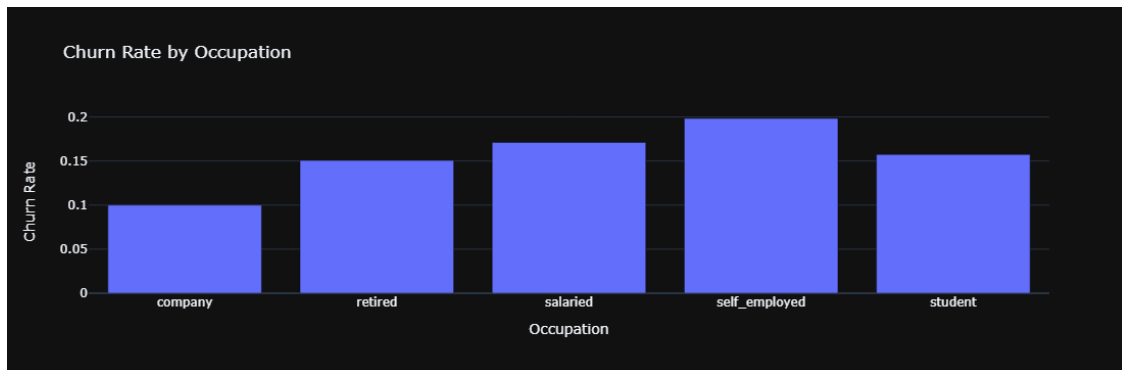
```
[10]: import plotly.express as px

# Assuming 'df' is your DataFrame with the 'age', 'current_balance', and 'churn'
#       ↪ columns
fig = px.scatter(df, x='age', y='current_balance',
                 color='churn', # Differentiate points by churn
                 title='Age vs Current Balance by Churn Status',
                 labels={'current_balance': 'Current Balance', 'age': 'Age'},
                 hover_data=['customer_id']) # Show customer_id on hover

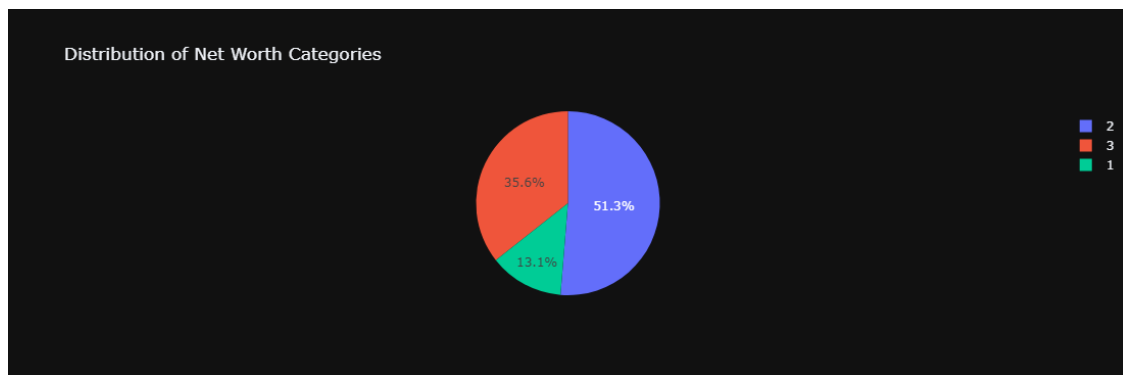
fig.update_layout(transition_duration=500)
fig.show()
```



```
[13]: fig = px.bar(df.groupby('occupation')['churn'].mean().reset_index(),
                  x='occupation', y='churn',
                  title='Churn Rate by Occupation',
                  labels={'churn': 'Churn Rate', 'occupation': 'Occupation'})
fig.show()
```



```
[14]: fig = px.pie(df, names='customer_nw_category', title='Distribution of Net Worth_
      ↳Categories',
                  labels={'customer_nw_category': 'Customer Net Worth Category'})
fig.show()
```



```
[15]: import joblib
```

```
# Save the model to disk
```

```
joblib.dump(grid_search.best_estimator_, 'final_model.joblib')
```

```
[15]: ['final_model.joblib']
```

```
[ ]:
```