Churn Customer Prediction

September 13, 2025

1 Customer Churn Prediction

This notebook demonstrates how to build a **supervised machine learning model** to predict customer churn, an online food ordering provider.

Why churn prediction matters? - Acquiring a new customer is 5–7 times more expensive than retaining an existing one. - By predicting churn, can target discounts, loyalty campaigns, or personalized offers to customers **before they leave**.

1.1 Step 1: Import Libraries

[1]:
```
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
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier

from sklearn.metrics import classification_report, confusion_matrix,␣
 ↪roc_auc_score, roc_curve
```

1.2 Step 2: Create Sample Dataset

[2]:
```
# Simulated dataset
np.random.seed(42)
n = 1000

data = pd.DataFrame({
    'customer_id': range(1, n+1),
    'last_order_days_ago': np.random.randint(1, 365, n),
    'total_orders': np.random.poisson(10, n),
    'avg_order_value': np.random.uniform(5, 50, n),
    'complaints': np.random.binomial(5, 0.1, n),
    'discount_used': np.random.randint(0, 20, n),
```

```
    'churn': np.random.binomial(1, 0.3, n)  # 1 = churned, 0 = active
})

data.head()
```

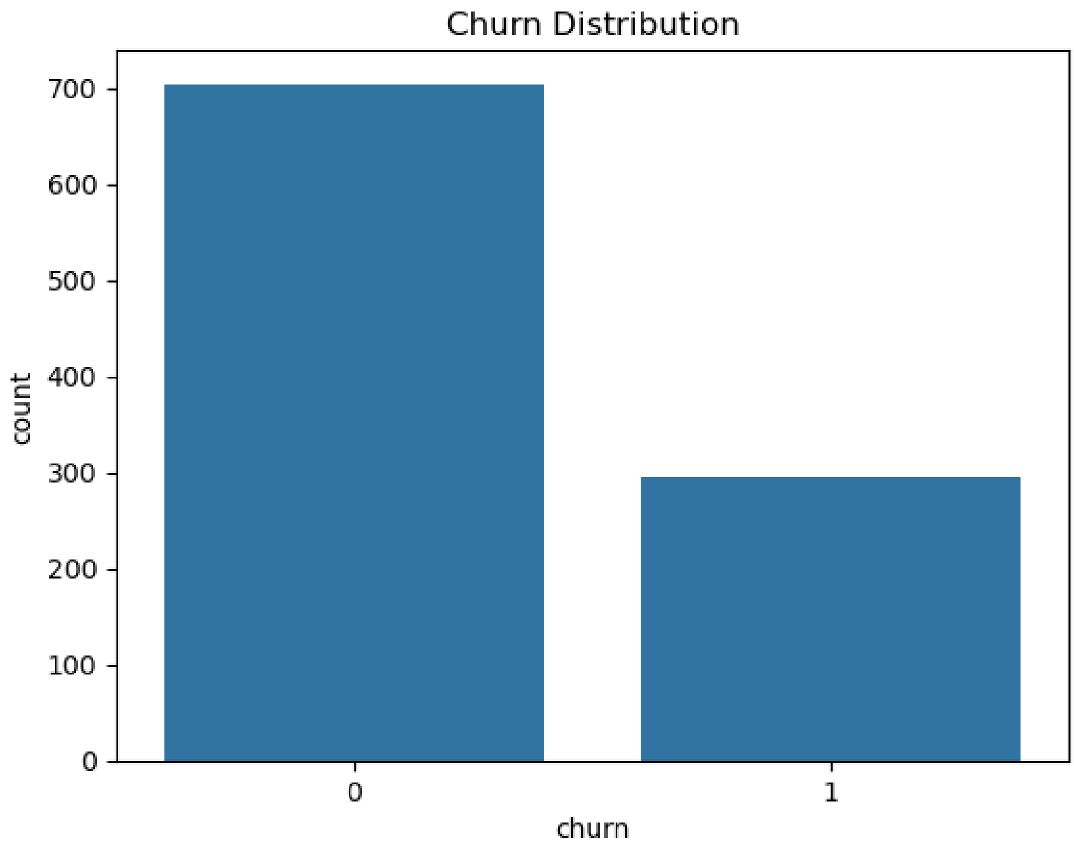
[2]:
```
   customer_id  last_order_days_ago  total_orders  avg_order_value  \
0            1                  103            13        24.118288
1            2                  349            11        45.852733
2            3                  271             9        27.817684
3            4                  107            18        13.456417
4            5                   72             7         8.463725

   complaints  discount_used  churn
0           1              6      0
1           0             17      0
2           2              6      0
3           0             16      1
4           1              6      0
```

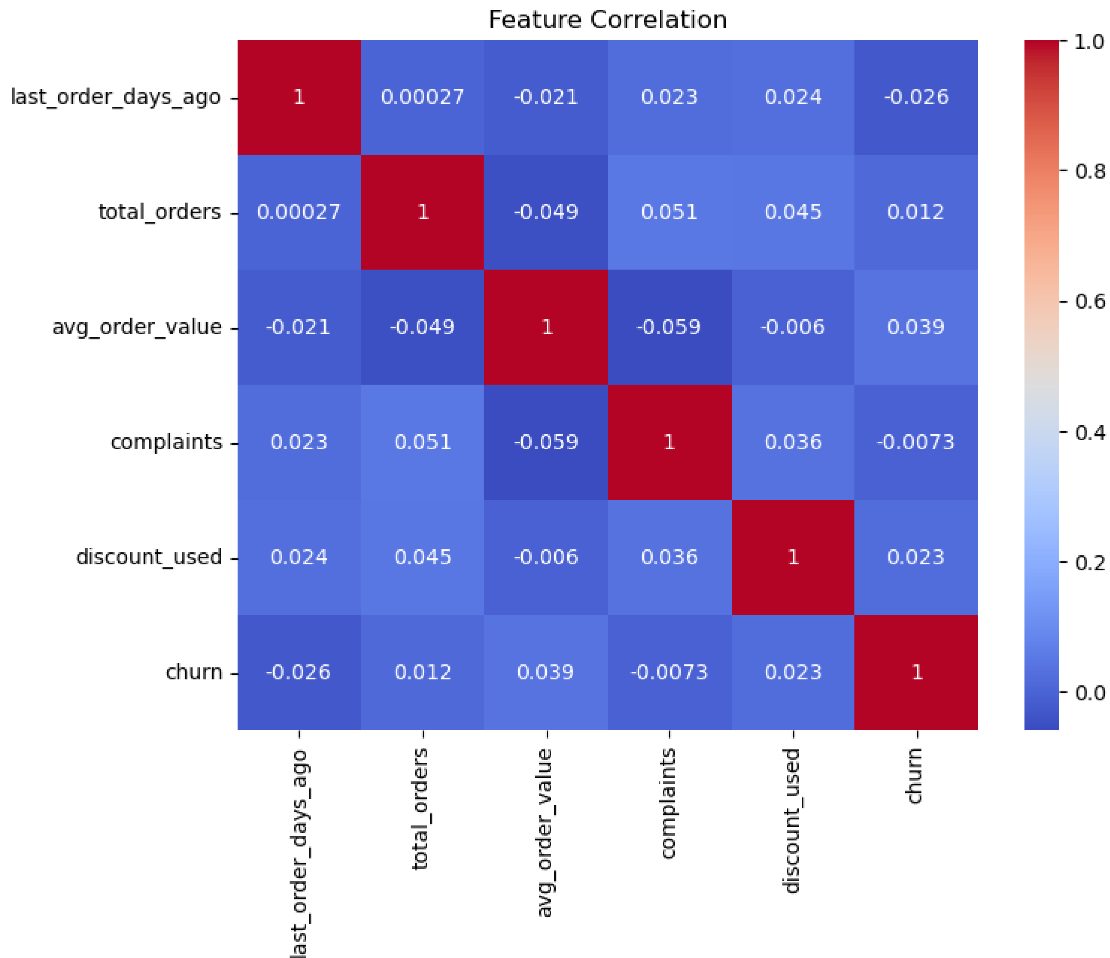
1.3 Step 3: Exploratory Data Analysis

[3]:
```
# Basic statistics
print(data.describe())

# Churn distribution
sns.countplot(x='churn', data=data)
plt.title("Churn Distribution")
plt.show()

# Correlation heatmap
plt.figure(figsize=(8,6))
sns.heatmap(data.drop('customer_id', axis=1).corr(), annot=True, cmap='coolwarm')
plt.title("Feature Correlation")
plt.show()
```

```
       customer_id  last_order_days_ago  total_orders  avg_order_value  \
count  1000.000000          1000.000000   1000.000000      1000.000000
mean    500.500000           181.374000     10.064000        27.272050
std     288.819436           103.360018      3.204399        12.945629
min       1.000000             1.000000      2.000000         5.001382
25%     250.750000            97.750000      8.000000        16.227675
50%     500.500000           180.000000     10.000000        27.098596
75%     750.250000           268.000000     12.000000        38.286977
max    1000.000000           364.000000     22.000000        49.980097

        complaints  discount_used        churn
count  1000.000000    1000.000000  1000.000000
mean      0.484000       9.375000     0.296000
```

std	0.675426	5.827034	0.456719
min	0.000000	0.000000	0.000000
25%	0.000000	4.000000	0.000000
50%	0.000000	9.000000	0.000000
75%	1.000000	15.000000	1.000000
max	3.000000	19.000000	1.000000





1.4 Step 4: Data Preprocessing

[4]:
```
X = data.drop(['customer_id', 'churn'], axis=1)
y = data['churn']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42, stratify=y)

# Feature scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

1.5 Step 5: Train Models

[5]:
```
# Logistic Regression
log_reg = LogisticRegression(max_iter=1000)
log_reg.fit(X_train_scaled, y_train)
y_pred_log = log_reg.predict(X_test_scaled)

# Random Forest
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)

# XGBoost
xgb = XGBClassifier(use_label_encoder=False, eval_metric='logloss',␣
 ↪random_state=42)
xgb.fit(X_train, y_train)
y_pred_xgb = xgb.predict(X_test)
```

```
C:\Users\SPINO SHOP\anaconda3\Lib\site-packages\xgboost\training.py:183:
UserWarning: [12:44:55] WARNING: C:\actions-
runner\_work\xgboost\xgboost\src\learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
```

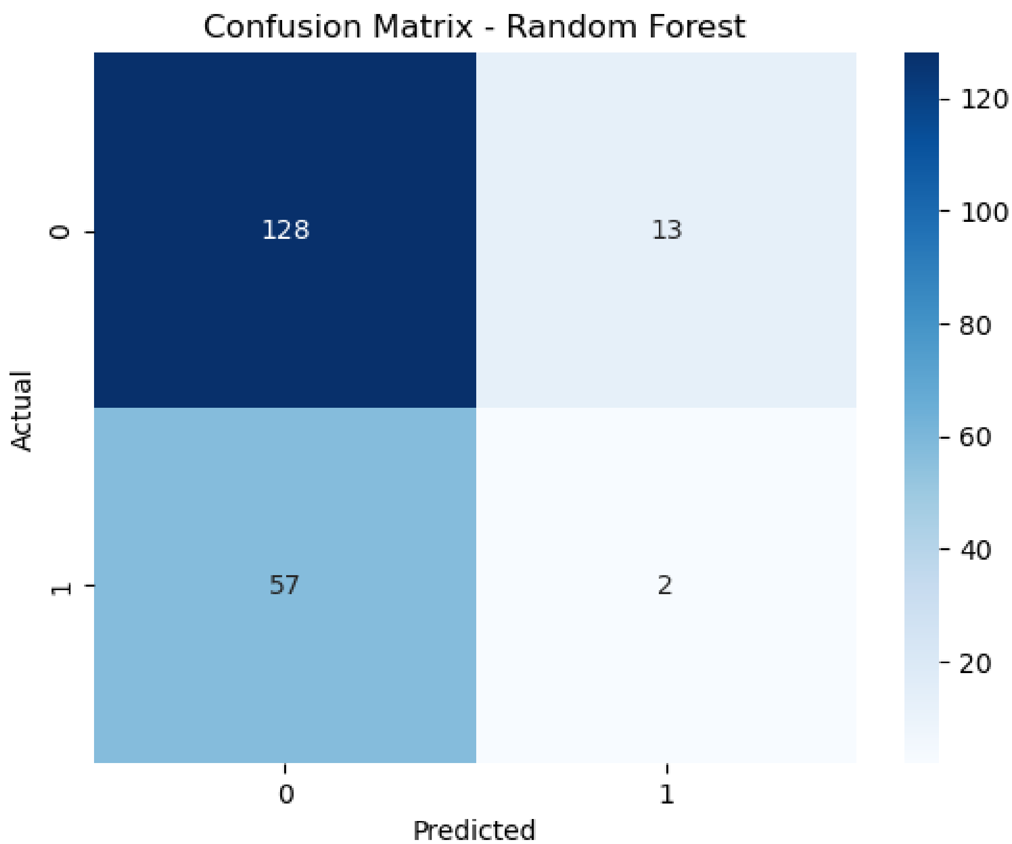
1.6 Step 6: Model Evaluation

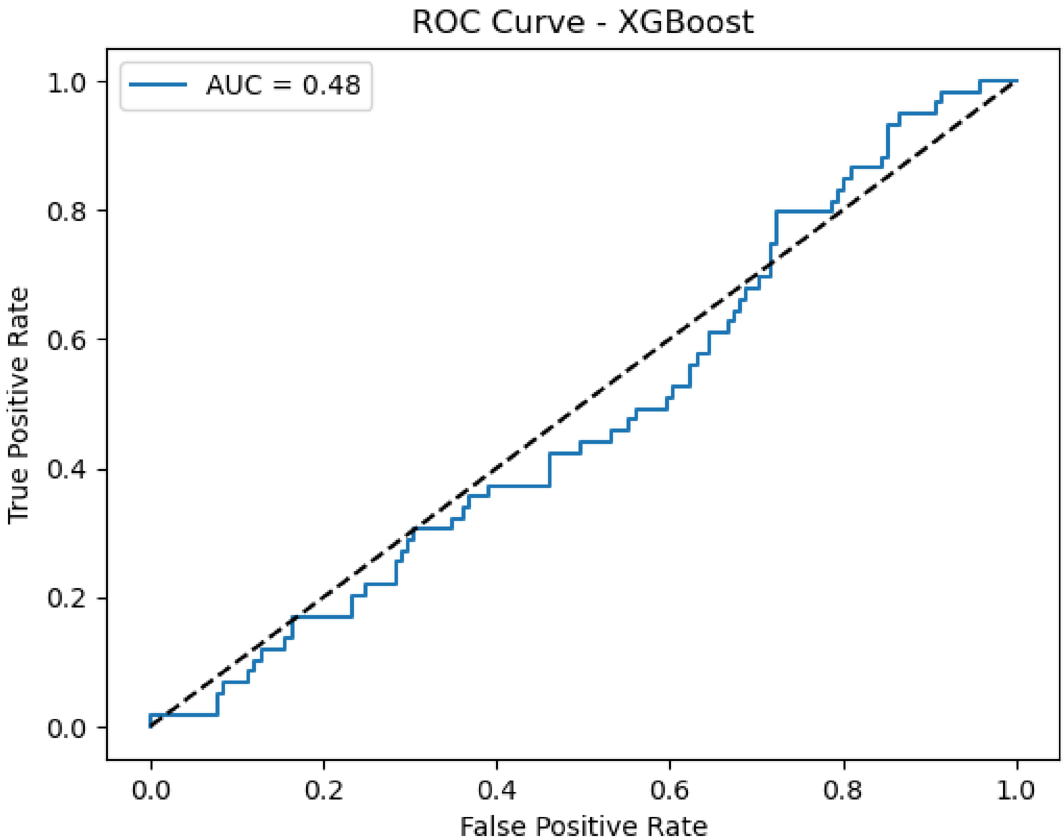
[6]:
```
print("Logistic Regression:\n", classification_report(y_test, y_pred_log))
print("Random Forest:\n", classification_report(y_test, y_pred_rf))
print("XGBoost:\n", classification_report(y_test, y_pred_xgb))

# Confusion Matrix for Random Forest
sns.heatmap(confusion_matrix(y_test, y_pred_rf), annot=True, fmt='d',␣
 ↪cmap='Blues')
plt.title("Confusion Matrix - Random Forest")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

# ROC Curve for XGBoost
y_probs = xgb.predict_proba(X_test)[:,1]
fpr, tpr, _ = roc_curve(y_test, y_probs)
plt.plot(fpr, tpr, label=f"AUC = {roc_auc_score(y_test, y_probs):.2f}")
plt.plot([0,1], [0,1], 'k--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve - XGBoost")
plt.legend()
```

```
plt.show()
```

```
C:\Users\SPINO SHOP\anaconda3\Lib\site-
packages\sklearn\metrics\_classification.py:1565: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\SPINO SHOP\anaconda3\Lib\site-
packages\sklearn\metrics\_classification.py:1565: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\SPINO SHOP\anaconda3\Lib\site-
packages\sklearn\metrics\_classification.py:1565: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

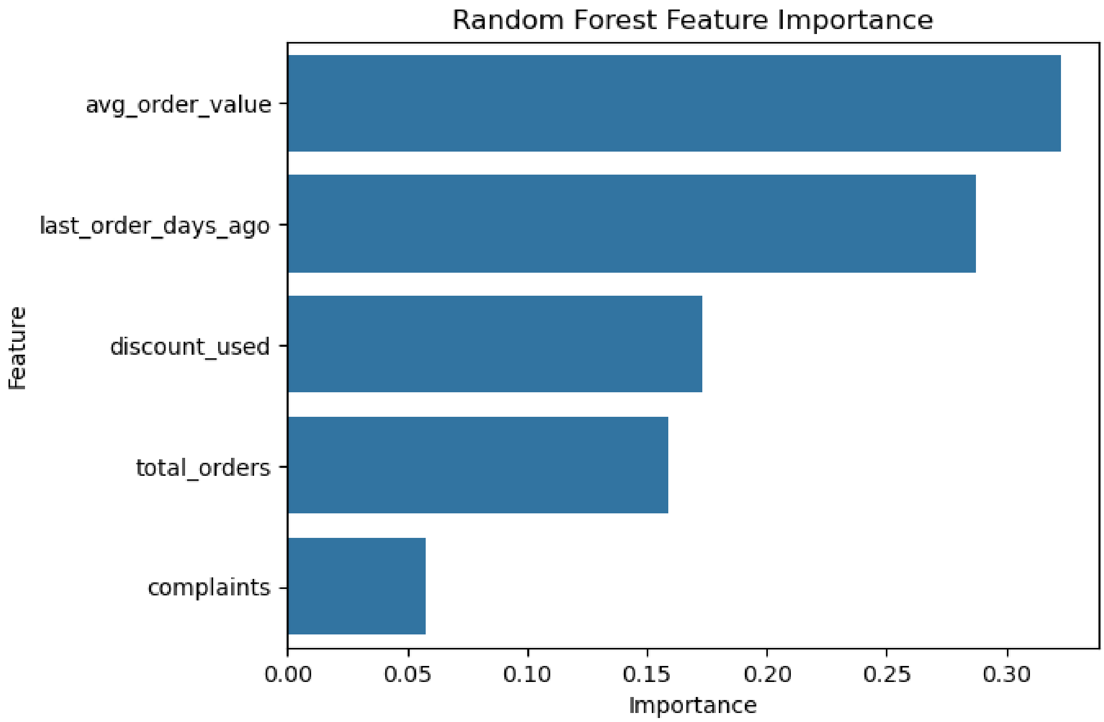
```
Logistic Regression:
              precision    recall  f1-score   support

           0       0.70      1.00      0.83       141
           1       0.00      0.00      0.00        59

    accuracy                           0.70       200
   macro avg       0.35      0.50      0.41       200
weighted avg       0.50      0.70      0.58       200

Random Forest:
              precision    recall  f1-score   support

           0       0.69      0.91      0.79       141
           1       0.13      0.03      0.05        59

    accuracy                           0.65       200
   macro avg       0.41      0.47      0.42       200
weighted avg       0.53      0.65      0.57       200

XGBoost:
              precision    recall  f1-score   support

           0       0.69      0.75      0.72       141
           1       0.26      0.20      0.23        59

    accuracy                           0.59       200
   macro avg       0.47      0.48      0.47       200
weighted avg       0.56      0.59      0.58       200
```


Confusion Matrix - Random Forest
Actual
0
1
128
13
57
2
0
1
Predicted
120
100
80
60
40
20



1.7 Step 7: Feature Importance

[7]:
```
importances = rf.feature_importances_
feat_names = X.columns

feat_imp = pd.DataFrame({'Feature': feat_names, 'Importance': importances})
feat_imp.sort_values(by='Importance', ascending=False, inplace=True)

sns.barplot(x='Importance', y='Feature', data=feat_imp)
plt.title("Random Forest Feature Importance")
plt.show()
```



1.8 Step 8: Conclusion & Business Use Case

- The models give us a way to predict which customers are likely to churn.
- We can integrate this model into the CRM/marketing system:
  - Send targeted discounts to **high-risk churners**.
  - Recommend personalized meals to re-engage customers.
  - Monitor churn rates across restaurants and regions.

Next Steps: - Replace synthetic dataset with real customer order data.
- Regularly retrain the model with new data.
- Deploy as an API to integrate with the app.