Telco Customer Churn Prediction Report

Author: Thattaphol Puttawithee

Tools: Python (Pandas, Scikit-learn, XGBoost), Power BI, SQL Server

Dataset: Telco Customer Churn (Kaggle)

Date: October 2025

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

Import data to Jupyter Notebook

Query data from SQL Server

```
import pandas as pd
from sqlalchemy import create_engine

pd.set_option('display.max_columns', None)

engine = create_engine(
    r"mssql+pyodbc://LAPTOP-QC1AHOCH\SQLEXPRESS/telco_db"
    r"?driver=ODBC+Driver+17+for+SQL+Server&trusted_connection=yes"
)

df = pd.read_sql("SELECT * FROM stg_Churn;", engine)
df
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	Strea
0	0002-ORFBO	Female	0	Yes	Yes	9	Yes	No	DSL	No	Yes	No	Yes	
1	0003-MKNFE	Male	0	No	No	9	Yes	Yes	DSL	No	No	No	No	
2	0004-TLHLJ	Male	0	No	No	4	Yes	No	Fiber optic	No	No	Yes	No	
3	0011-IGKFF	Male	1	Yes	No	13	Yes	No	Fiber optic	No	Yes	Yes	No	
4	0013-EXCHZ	Female	1	Yes	No	3	Yes	No	Fiber optic	No	No	No	Yes	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
7038	9987-LUTYD	Female	0	No	No	13	Yes	No	DSL	Yes	No	No	Yes	
7039	9992-RRAMN	Male	0	Yes	No	22	Yes	Yes	Fiber optic	No	No	No	No	
7040	9992-UJOEL	Male	0	No	No	2	Yes	No	DSL	No	Yes	No	No	

```
df.info()
df.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   customerID        7043 non-null   object
 1   gender            7043 non-null   object
 2   SeniorCitizen     7043 non-null   object
 3   Partner           7043 non-null   object
 4   Dependents        7043 non-null   object
 5   tenure            7043 non-null   int64
 6   PhoneService      7043 non-null   object
 7   MultipleLines     7043 non-null   object
 8   InternetService   7043 non-null   object
 9   OnlineSecurity    7043 non-null   object
 10  OnlineBackup      7043 non-null   object
 11  DeviceProtection  7043 non-null   object
 12  TechSupport       7043 non-null   object
 13  StreamingTV       7043 non-null   object
 14  StreamingMovies   7043 non-null   object
 15  Contract          7043 non-null   object
 16  PaperlessBilling  7043 non-null   object
 17  PaymentMethod     7043 non-null   object
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7032 non-null   float64
 20  Churn             7043 non-null   object
dtypes: float64(2), int64(1), object(18)
memory usage: 1.1+ MB
```

	tenure	MonthlyCharges	TotalCharges
count	7043.000000	7043.000000	7032.000000
mean	32.371149	64.761692	2283.300441
std	24.559481	30.090047	2266.771363
min	0.000000	18.250000	18.799999
25%	9.000000	35.500000	401.449997
50%	29.000000	70.349998	1397.475037
75%	55.000000	89.849998	3794.737488
max	72.000000	118.750000	8684.799805

Check NULL

Check null value in the data

```
df.isna().sum()
```

```
customerID           0
gender               0
SeniorCitizen        0
Partner              0
Dependents           0
tenure               0
PhoneService         0
MultipleLines        0
InternetService      0
OnlineSecurity       0
OnlineBackup         0
DeviceProtection     0
TechSupport          0
StreamingTV          0
StreamingMovies      0
Contract             0
PaperlessBilling     0
PaymentMethod        0
MonthlyCharges       0
TotalCharges        11
Churn                0
dtype: int64
```

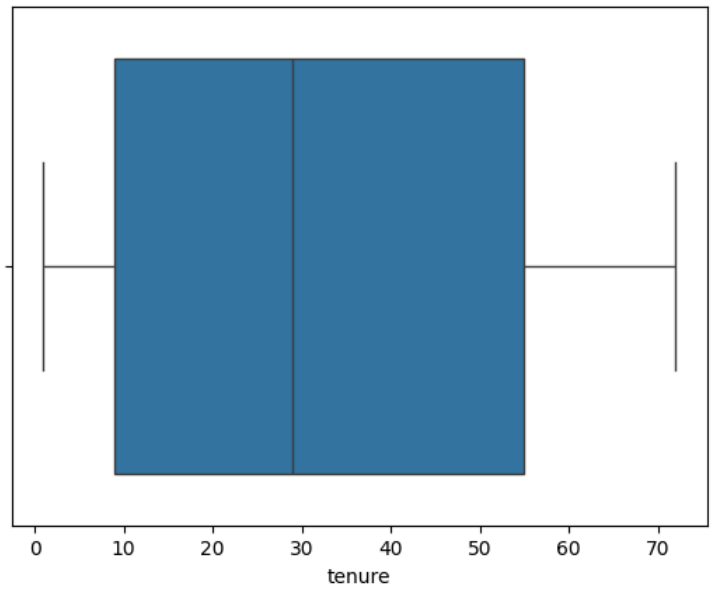
Remove null

```
df = df.dropna()
```

Check Outliers

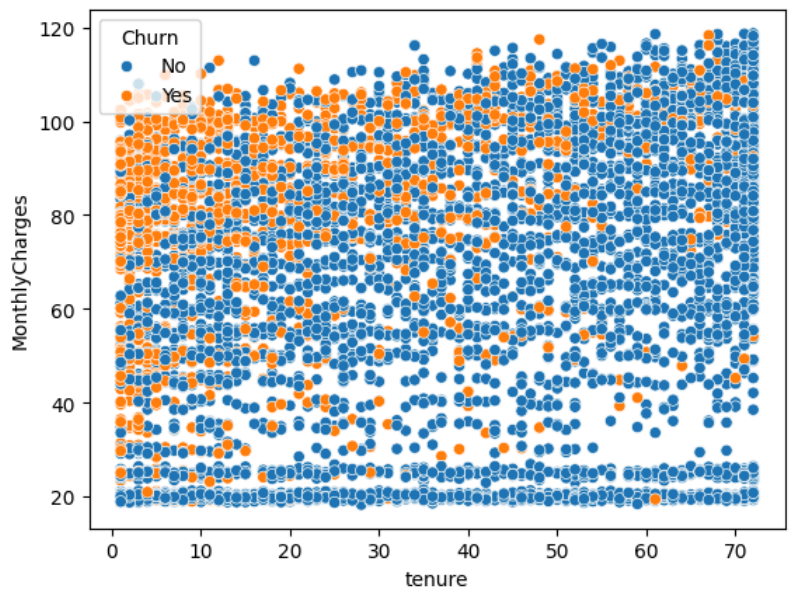
```
sns.boxplot(x=df['tenure'])
```

```
<Axes: xlabel='tenure'>
```




```
sns.scatterplot(x='tenure', y='MonthlyCharges', hue='Churn', data=df)
```

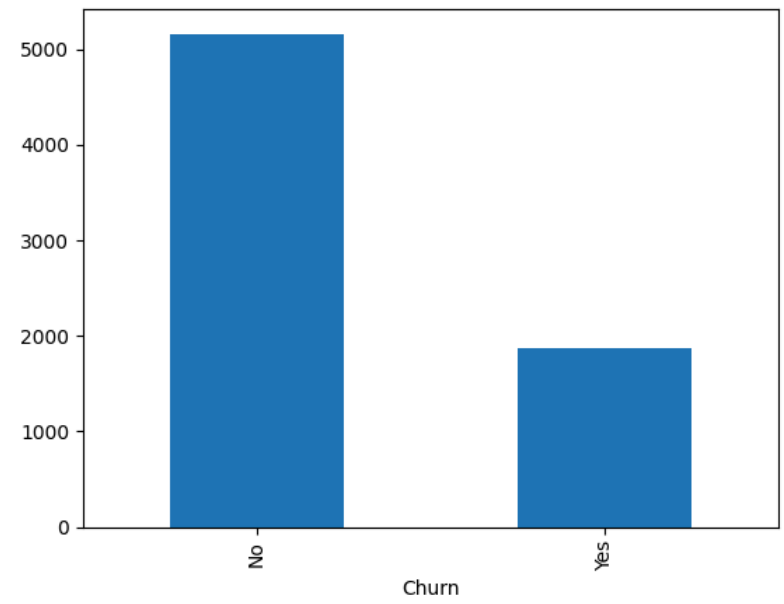
```
<Axes: xlabel='tenure', ylabel='MonthlyCharges'>
```



Check data balance

```
df['Churn'].value_counts().plot(kind='bar')
```

<Axes: xlabel='Churn'>



df

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	Strea
0	0002-ORFBO	Female	0	Yes	Yes	9	Yes	No	DSL	No	Yes	No	Yes	
1	0003-MKNFE	Male	0	No	No	9	Yes	Yes	DSL	No	No	No	No	
2	0004-TLHLJ	Male	0	No	No	4	Yes	No	Fiber optic	No	No	Yes	No	
3	0011-IGKFF	Male	1	Yes	No	13	Yes	No	Fiber optic	No	Yes	Yes	No	
4	0013-EXCHZ	Female	1	Yes	No	3	Yes	No	Fiber optic	No	No	No	Yes	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
7038	9987-LUTYD	Female	0	No	No	13	Yes	No	DSL	Yes	No	No	Yes	
7039	9992-RRAMN	Male	0	Yes	No	22	Yes	Yes	Fiber optic	No	No	No	No	
7040	9992-UJOEL	Male	0	No	No	2	Yes	No	DSL	No	Yes	No	No	

```
binary_cols = ['Partner', 'Dependents', 'PhoneService', 'MultipleLines',
               'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
               'TechSupport', 'StreamingTV', 'StreamingMovies',
               'PaperlessBilling', 'Churn']

df[binary_cols] = df[binary_cols].replace({'Yes': 1, 'No': 0})
df['gender'] = df['gender'].replace({'Male': 1, 'Female': 0})
df['MultipleLines'] = df['MultipleLines'].replace({'No phone service': -1})
df
```

```
C:\Temp\ipykernel_13568\2260154420.py:6: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed in a future version. To retain the old behavior, explicitly call `result.infer_objects(copy=False)
  df[binary_cols] = df[binary_cols].replace({'Yes': 1, 'No': 0})
C:\Temp\ipykernel_13568\2260154420.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[binary_cols] = df[binary_cols].replace({'Yes': 1, 'No': 0})
C:\Temp\ipykernel_13568\2260154420.py:7: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed in a future version. To retain the old behavior, explicitly call `result.infer_objects(copy=False)
  df['gender'] = df['gender'].replace({'Male': 1, 'Female': 0})
C:\Temp\ipykernel_13568\2260154420.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['gender'] = df['gender'].replace({'Male': 1, 'Female': 0})
C:\Temp\ipykernel_13568\2260154420.py:8: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed in a future version. To retain the old behavior, explicitly call `result.infer_objects(copy=False)
  df['MultipleLines'] = df['MultipleLines'].replace({'No phone service': -1})
C:\Temp\ipykernel_13568\2260154420.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['MultipleLines'] = df['MultipleLines'].replace({'No phone service': -1})
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	Strea
0	0002-ORFBO	0	0	1	1	9	1	0	DSL	0	1	0	1	
1	0003-MKNFE	1	0	0	0	9	1	1	DSL	0	0	0	0	
2	0004-TLHLJ	1	0	0	0	4	1	0	Fiber optic	0	0	1	0	
3	0011-IGKFF	1	1	1	0	13	1	0	Fiber optic	0	1	1	0	
4	0013-EXCHZ	0	1	1	0	3	1	0	Fiber optic	0	0	0	1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
7038	9987-LUTYD	0	0	0	0	13	1	0	DSL	1	0	0	1	
7039	9992-RRAMN	1	0	1	0	22	1	1	Fiber optic	0	0	0	0	
7040	9992-UJOEL	1	0	0	0	2	1	0	DSL	0	1	0	0	

```
Services_cols = (['OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies'])
df[Services_cols] = df[Services_cols].replace({'No internet service': 0})
df.isin(['No internet service']).sum()
```

```
C:\Temp\ipykernel_13568\3187367540.py:2: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed in a future version. To retain the old behavior, explicitly call `result.infer_objects(copy=False)
  df[Services_cols] = df[Services_cols].replace({'No internet service': 0})
```

```
C:\Temp\ipykernel_13568\3187367540.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[Services_cols] = df[Services_cols].replace({'No internet service': 0})
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
```

```
df['SeniorCitizen'] = df['SeniorCitizen'].astype(int)
```

```
C:\Temp\ipykernel_13568\2584085859.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['SeniorCitizen'] = df['SeniorCitizen'].astype(int)
```

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
numeric_cols = ['tenure', 'MonthlyCharges', 'TotalCharges']

df[numeric_cols] = scaler.fit_transform(df[numeric_cols])
df
```

```
C:\Temp\ipykernel_13568\454533415.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[numeric_cols] = scaler.fit_transform(df[numeric_cols])
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	Str
0	0002-ORFBO	0	0	1	1	-0.954296	1	0	DSL	0	1	0	1	
1	0003-MKNFE	1	0	0	0	-0.954296	1	1	DSL	0	0	0	0	
2	0004-TLHLJ	1	0	0	0	-1.158016	1	0	Fiber optic	0	0	1	0	
3	0011-IGKFF	1	1	1	0	-0.791321	1	0	Fiber optic	0	1	1	0	
4	0013-EXCHZ	0	1	1	0	-1.198760	1	0	Fiber optic	0	0	0	1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
7038	9987-LUTYD	0	0	0	0	-0.791321	1	0	DSL	1	0	0	1	
7039	9992-RRAMN	1	0	1	0	-0.424625	1	1	Fiber optic	0	0	0	0	
7040	9992-UJOEL	1	0	0	0	-1.239504	1	0	DSL	0	1	0	0	

```
categorical_cols = ['InternetService', 'Contract', 'PaymentMethod']

df = pd.get_dummies(df, columns=categorical_cols, drop_first=True, dtype=int)
df
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	Strea
0	0002-ORFBO	0	0	1	1	-0.954296	1	0	0	1	0	1	1	
1	0003-MKNFE	1	0	0	0	-0.954296	1	1	0	0	0	0	0	
2	0004-TLHLJ	1	0	0	0	-1.158016	1	0	0	0	1	0	0	
3	0011-IGKFF	1	1	1	0	-0.791321	1	0	0	1	1	0	1	
4	0013-EXCHZ	0	1	1	0	-1.198760	1	0	0	0	0	1	1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
7038	9987-LUTYD	0	0	0	0	-0.791321	1	0	1	0	0	1	0	
7039	9992-RRAMN	1	0	1	0	-0.424625	1	1	0	0	0	0	0	
7040	9992-UJOEL	1	0	0	0	-1.239504	1	0	0	1	0	0	0	
7041	9993-LHIEB	1	0	1	1	1.408853	1	0	1	0	1	1	0	
7042	9995-HOTOH	1	0	1	1	1.245878	0	-1	1	1	1	0	1	

7032 rows × 25 columns

Train Test split

```
from sklearn.model_selection import train_test_split
X = df.drop(['Churn', 'customerID'], 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, stratify=y
```

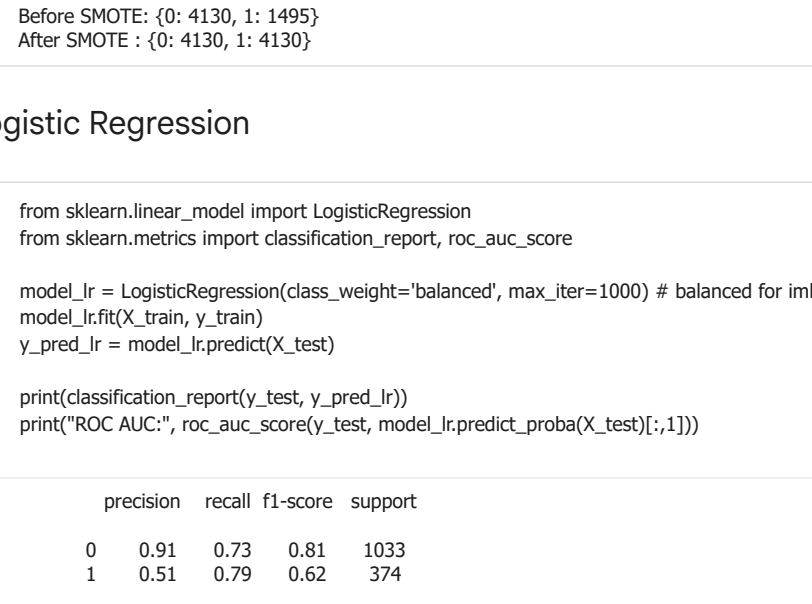
```
)

X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((5625, 23), (1407, 23), (5625,), (1407,))
```

Class imbalance

```
y_train.value_counts().plot(kind='bar')
```

```
<Axes: xlabel='Churn'>
```



Prepare SMOTE

```
import pandas as pd
from imblearn.over_sampling import SMOTE

sm = SMOTE(random_state=42)
X_train_res, y_train_res = sm.fit_resample(X_train, y_train)

y_train_series = pd.Series(y_train)
y_train_res_series = pd.Series(y_train_res)

print("Before SMOTE:", y_train_series.value_counts().to_dict())
print("After SMOTE :", y_train_res_series.value_counts().to_dict())
```

```
Before SMOTE: {0: 4130, 1: 1495}
After SMOTE : {0: 4130, 1: 4130}
```

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, roc_auc_score

model_lr = LogisticRegression(class_weight='balanced', max_iter=1000) # balanced for imbalanced data
model_lr.fit(X_train, y_train)
y_pred_lr = model_lr.predict(X_test)

print(classification_report(y_test, y_pred_lr))
print("ROC AUC:", roc_auc_score(y_test, model_lr.predict_proba(X_test)[:,1]))
```

```
              precision    recall  f1-score   support

           0       0.91      0.73      0.81      1033
           1       0.51      0.79      0.62       374

    accuracy                           0.74      1407
   macro avg       0.71      0.76      0.71      1407
weighted avg       0.80      0.74      0.76      1407

ROC AUC: 0.8447178924372706
```

RandomForest

```
from sklearn.ensemble import RandomForestClassifier

model_rf = RandomForestClassifier(
    n_estimators=300,
    class_weight='balanced_subsample', # balanced_subsample for imbalanced data
    max_depth=10,
    random_state=42
)
model_rf.fit(X_train, y_train)
rf_pred = model_rf.predict(X_test)

print("ROC AUC:", roc_auc_score(y_test, model_rf.predict_proba(X_test)[:,1]))
print(classification_report(y_test, rf_pred))
```

```
ROC AUC: 0.8447101272965404
              precision    recall  f1-score   support

           0       0.89      0.79      0.84      1033
           1       0.56      0.72      0.63       374

    accuracy                           0.78      1407
   macro avg       0.72      0.76      0.74      1407
weighted avg       0.80      0.78      0.78      1407
```

XGB

```
from xgboost import XGBClassifier

xgb = XGBClassifier(
    scale_pos_weight=(len(y_train[y_train==0]) / len(y_train[y_train==1])),
    learning_rate=0.05,
    n_estimators=500,
    max_depth=4,
    subsample=0.8,
    colsample_bytree=0.8,
    random_state=42
)
xgb.fit(X_train, y_train)
xgb_pred = xgb.predict(X_test)

print("ROC AUC:", roc_auc_score(y_test, xgb.predict_proba(X_test)[:,1]))
print("\nClassification Report:\n", classification_report(y_test, xgb_pred))
```

```
ROC AUC: 0.8406166039415854

Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.76      0.82      1033
           1       0.53      0.76      0.63       374

    accuracy                           0.76      1407
   macro avg       0.71      0.76      0.72      1407
weighted avg       0.80      0.76      0.77      1407
```

Hyperparameter Tuning

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import RandomizedSearchCV, StratifiedKFold
from scipy.stats import loguniform

cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

lr = LogisticRegression(class_weight='balanced', solver='liblinear', max_iter=5000)

lr_param = {
    "C": loguniform(1e-3, 1e2),   # 0.001..100
    "penalty": ["l1", "l2"],
}

lr_rs = RandomizedSearchCV(
    lr, lr_param, n_iter=40, scoring="roc_auc", cv=cv,
    n_jobs=-1, random_state=42, error_score="raise"
)
lr_rs.fit(X_train, y_train)
print("Best parameters:", lr_rs.best_params_)
```

```
Best parameters: {'C': np.float64(3.4702669886504163), 'penalty': 'l2'}
```

```
import numpy as np
from sklearn.metrics import precision_score, recall_score, f1_score, roc_auc_score, classification_report

best_lr = lr_rs.best_estimator_
y_proba = best_lr.predict_proba(X_test)[:, 1]

for t in [0.3, 0.4, 0.5, 0.6, 0.7]:
    y_pred_t = (y_proba >= t).astype(int)
    precision = precision_score(y_test, y_pred_t)
    recall = recall_score(y_test, y_pred_t)
    f1 = f1_score(y_test, y_pred_t)
    print(f"Threshold = {t:.1f} | Precision = {precision:.3f} | Recall = {recall:.3f} | F1 = {f1:.3f}")
```

```
Threshold = 0.3 | Precision = 0.426 | Recall = 0.920 | F1 = 0.582
Threshold = 0.4 | Precision = 0.465 | Recall = 0.861 | F1 = 0.604
Threshold = 0.5 | Precision = 0.514 | Recall = 0.797 | F1 = 0.625
Threshold = 0.6 | Precision = 0.558 | Recall = 0.709 | F1 = 0.624
Threshold = 0.7 | Precision = 0.614 | Recall = 0.561 | F1 = 0.587
```

XMG

```
from xgboost import XGBClassifier
from sklearn.model_selection import RandomizedSearchCV, StratifiedKFold
from scipy.stats import randint, uniform, loguniform
from sklearn.metrics import roc_auc_score

cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# For XGBoost class imbalance
scale = len(y_train[y_train==0]) / len(y_train[y_train==1])

xgb = XGBClassifier(
    objective='binary:logistic',
    eval_metric='auc',
    scale_pos_weight=scale,
    random_state=42
)

xgb_param = {
    "n_estimators": randint(300, 800),
    "learning_rate": uniform(0.01, 0.2),
    "max_depth": randint(3, 7),
    "min_child_weight": randint(1, 8),
    "subsample": uniform(0.6, 0.4),
    "colsample_bytree": uniform(0.6, 0.4),
    "gamma": uniform(0, 0.5),
    "reg_lambda": loguniform(1e-2, 1e2),
    "reg_alpha": loguniform(1e-3, 1e1),
}

xgb_rs = RandomizedSearchCV(
```

```
    estimator=xgb,
    param_distributions=xgb_param,
    n_iter=50,
    scoring='roc_auc',
    cv=cv,
    n_jobs=-1,
    random_state=42,
    verbose=1
)

xgb_rs.fit(X_train, y_train)

print("Best parameters:", xgb_rs.best_params_)
```

```
Fitting 5 folds for each of 50 candidates, totalling 250 fits
Best parameters: {'colsample_bytree': np.float64(0.8232408008069365), 'gamma': np.float64(0.2019180855290204), 'learning_rate': np.float64(0.02297844942179631), 'max_depth': 4, 'min_child_weight': 4, 'n_estimators': 5
```

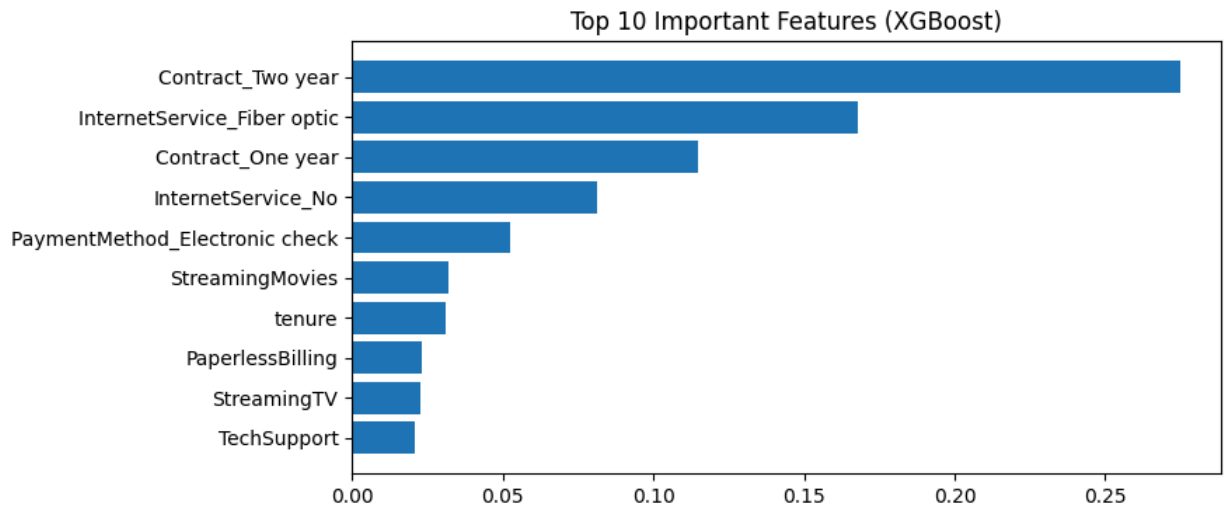
```
import numpy as np
from sklearn.metrics import precision_score, recall_score, f1_score, roc_auc_score, classification_report

best_xgb = xgb_rs.best_estimator_
y_proba = best_xgb.predict_proba(X_test)[:, 1]

for t in [0.3, 0.4, 0.5, 0.6, 0.7]:
    y_pred_t = (y_proba >= t).astype(int)
    precision = precision_score(y_test, y_pred_t)
    recall = recall_score(y_test, y_pred_t)
    f1 = f1_score(y_test, y_pred_t)
    print(f"Threshold = {t:.1f} | Precision = {precision:.3f} | Recall = {recall:.3f} | F1 = {f1:.3f}")
```

```
Threshold = 0.3 | Precision = 0.444 | Recall = 0.906 | F1 = 0.596
Threshold = 0.4 | Precision = 0.499 | Recall = 0.856 | F1 = 0.631
Threshold = 0.5 | Precision = 0.534 | Recall = 0.775 | F1 = 0.632
Threshold = 0.6 | Precision = 0.581 | Recall = 0.703 | F1 = 0.636
Threshold = 0.7 | Precision = 0.624 | Recall = 0.545 | F1 = 0.582
```

```
# FINAL MODEL - XGBoost (threshold = 0.4)
import joblib
import numpy as np

final_xgb = xgb_rs.best_estimator_  # จาก RandomizedSearchCV
final_threshold = 0.4

# save model and threshold
joblib.dump(final_xgb, "final_xgb_model.joblib")
np.save("final_threshold.npy", np.array([final_threshold]))
print("Model and threshold saved.")
```

```
Model and threshold saved.
```

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

importance = pd.DataFrame({
    "Feature": X_train.columns,
    "Importance": final_xgb.feature_importances_
}).sort_values("Importance", ascending=False).head(10)

plt.figure(figsize=(8,4))
plt.barh(importance["Feature"], importance["Importance"])
plt.gca().invert_yaxis()
plt.title("Top 10 Important Features (XGBoost)")
plt.show()
```



Executive Summary

“Developed a customer churn prediction model using XGBoost, achieving 0.84 ROC-AUC. At threshold = 0.4, the model achieves 0.50 Precision, 0.86 Recall, and 0.63 F1-score. Insights show that contract type, tenure, and monthly charges are the strongest predictors of churn.”

- Key Insights
  - Long-term contracts ⟶ drastically reduce churn
  - Fiber optic customers ⟶ high churn risk
  - Low tenure + high monthly charges ⟶ most likely to churn
- Recommended Business Actions
  - Offer discounts to short-tenure customers
  - Improve Fiber optic service satisfaction
  - Retarget paperless billing users with higher satisfaction campaigns