Task4

November 19, 2025

1 Task 4

[34]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import LabelEncoder, OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report,␣
↪confusion_matrix
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestClassifier
import joblib
```

[35]:
```
df = pd.read_csv("Telco-Customer-Churn.csv")
df.head()
```

[35]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	…	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling
0	7590-VHVEG	Female	0	Yes	No	1	No	…	No	No	No	No	Month-to-month	Yes
1	5575-GNVDE	Male	0	No	No	34	Yes	…	Yes	No	No	No	One year	No
2	3668-QPYBK	Male	0	No	No	2	Yes	…	No	No	No	No	Month-to-month	Yes
3	7795-CFOCW	Male	0	No	No	45	No	…	Yes	No	No	No	One year	No
4	9237-HQITU	Female	0	No	No	2	Yes	…	No	No	No	No	Month-to-month	Yes

	MultipleLines	InternetService	OnlineSecurity
0	No phone service	DSL	No
1	No	DSL	Yes
2	No	DSL	Yes
3	No phone service	DSL	Yes
4	No	Fiber optic	No

```
3         Yes          No              No        One year              No
4          No          No              No  Month-to-month             Yes

               PaymentMethod MonthlyCharges  TotalCharges Churn
0           Electronic check          29.85         29.85    No
1               Mailed check          56.95        1889.5    No
2               Mailed check          53.85        108.15   Yes
3  Bank transfer (automatic)          42.30       1840.75    No
4           Electronic check          70.70        151.65   Yes

[5 rows x 21 columns]
```

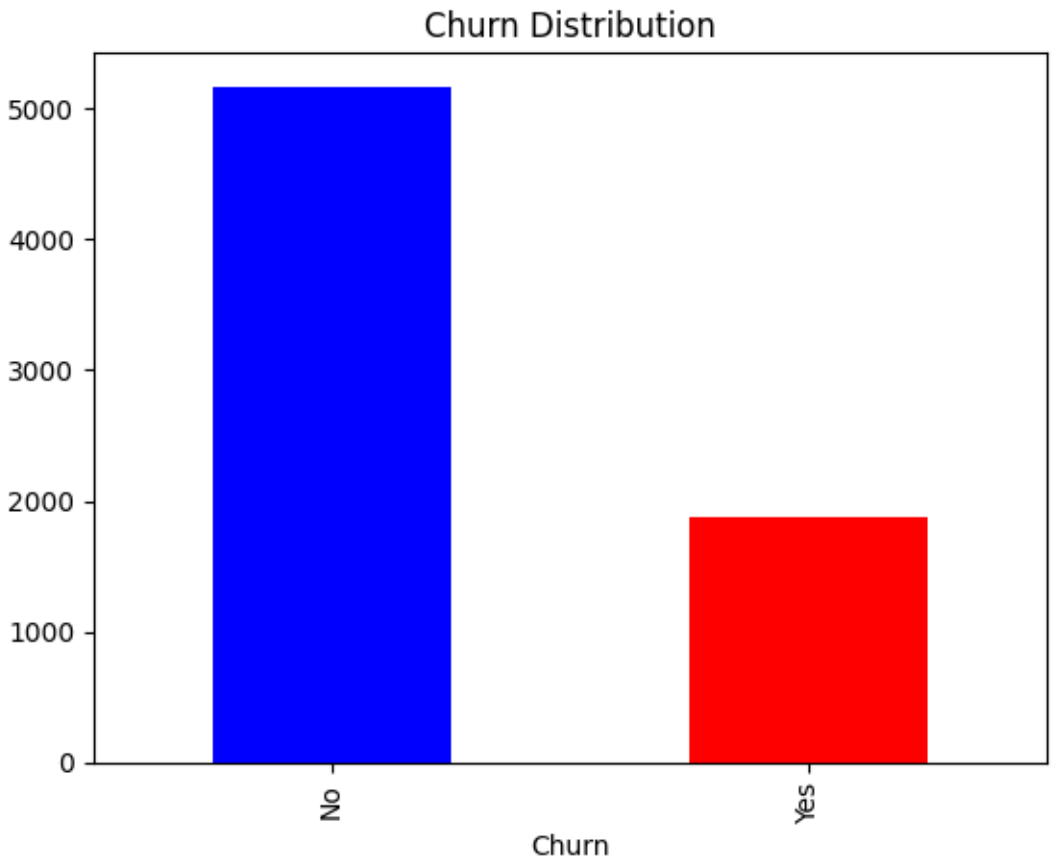
[36]:
```
print(df.shape)
df.info()
df.describe()
```

```
(7043, 21)
<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   int64
 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      7043 non-null   object
 20  Churn             7043 non-null   object
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
```

[36]:

	SeniorCitizen	tenure	MonthlyCharges
count	7043.000000	7043.000000	7043.000000
mean	0.162147	32.371149	64.761692
std	0.368612	24.559481	30.090047
min	0.000000	0.000000	18.250000
25%	0.000000	9.000000	35.500000
50%	0.000000	29.000000	70.350000
75%	0.000000	55.000000	89.850000
max	1.000000	72.000000	118.750000

[37]:
```
df["Churn"].value_counts().plot(kind="bar", color=['blue','red'])
plt.title("Churn Distribution")
plt.show()
```



[38]:
```
df = df.replace(" ", np.nan)
df = df.dropna()
df.isnull().sum()
```

[38]:
```
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
```

[39]:
```
df["TotalCharges"] = pd.to_numeric(df["TotalCharges"])

X = df.drop("Churn", axis=1)
y = df["Churn"]

le = LabelEncoder()
y = le.fit_transform(y)
```

[40]:
```
categorical_cols = X.select_dtypes(include=["object"]).columns
numeric_cols = X.select_dtypes(include=["int64", "float64"]).columns
categorical_cols, numeric_cols
```

[40]:
```
(Index(['customerID', 'gender', 'Partner', 'Dependents', 'PhoneService',
        'MultipleLines', 'InternetService', 'OnlineSecurity', 'OnlineBackup',
        'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies',
        'Contract', 'PaperlessBilling', 'PaymentMethod'],
       dtype='object'),
 Index(['SeniorCitizen', 'tenure', 'MonthlyCharges', 'TotalCharges'],
dtype='object'))
```

[41]:
```
preprocess = ColumnTransformer(
    transformers=[
        ("cat", OneHotEncoder(handle_unknown='ignore'), categorical_cols),
        ("num", StandardScaler(), numeric_cols)
```

```
    ]
)
```

[42]:
```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
model = Pipeline(
    steps=[
        ("preprocess", preprocess),
        ("classifier", RandomForestClassifier(n_estimators=200,␣
 ↪random_state=42))
    ]
)
```

[43]:
```
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

[44]:
```
accuracy = accuracy_score(y_test, y_pred)
accuracy
```

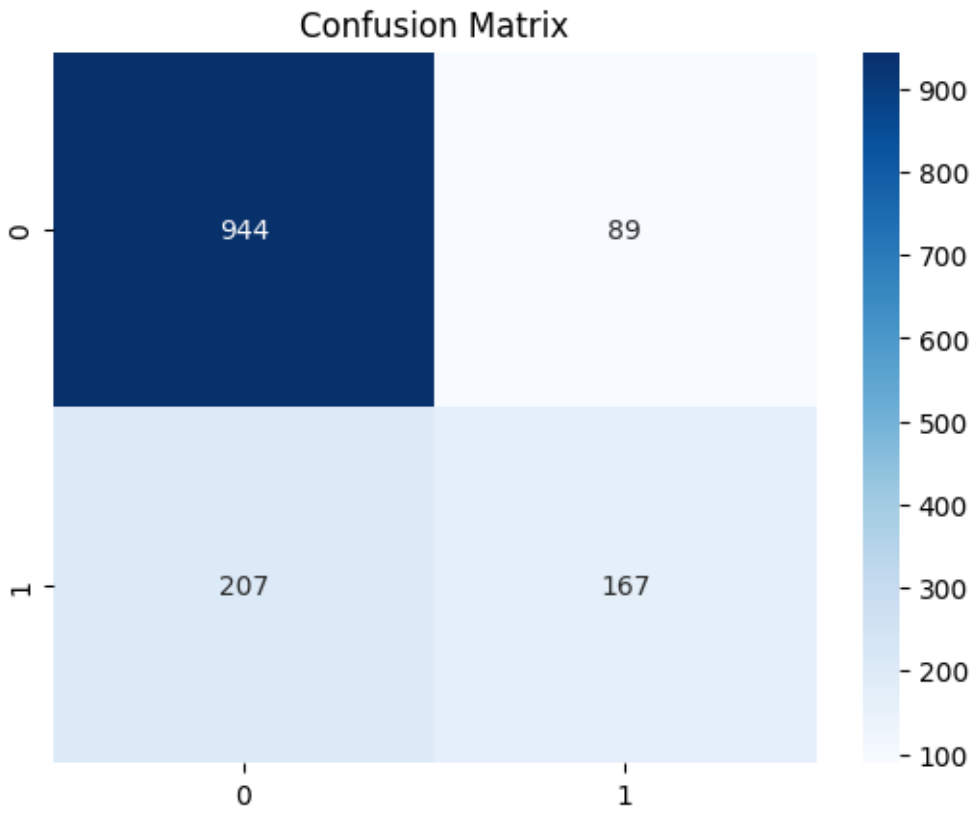
[44]:
```
0.7896233120113717
```

[45]:
```
print(classification_report(y_test, y_pred))
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix")
plt.show()
```

```
              precision    recall  f1-score   support

           0       0.82      0.91      0.86      1033
           1       0.65      0.45      0.53       374

    accuracy                           0.79      1407
   macro avg       0.74      0.68      0.70      1407
weighted avg       0.78      0.79      0.78      1407
```



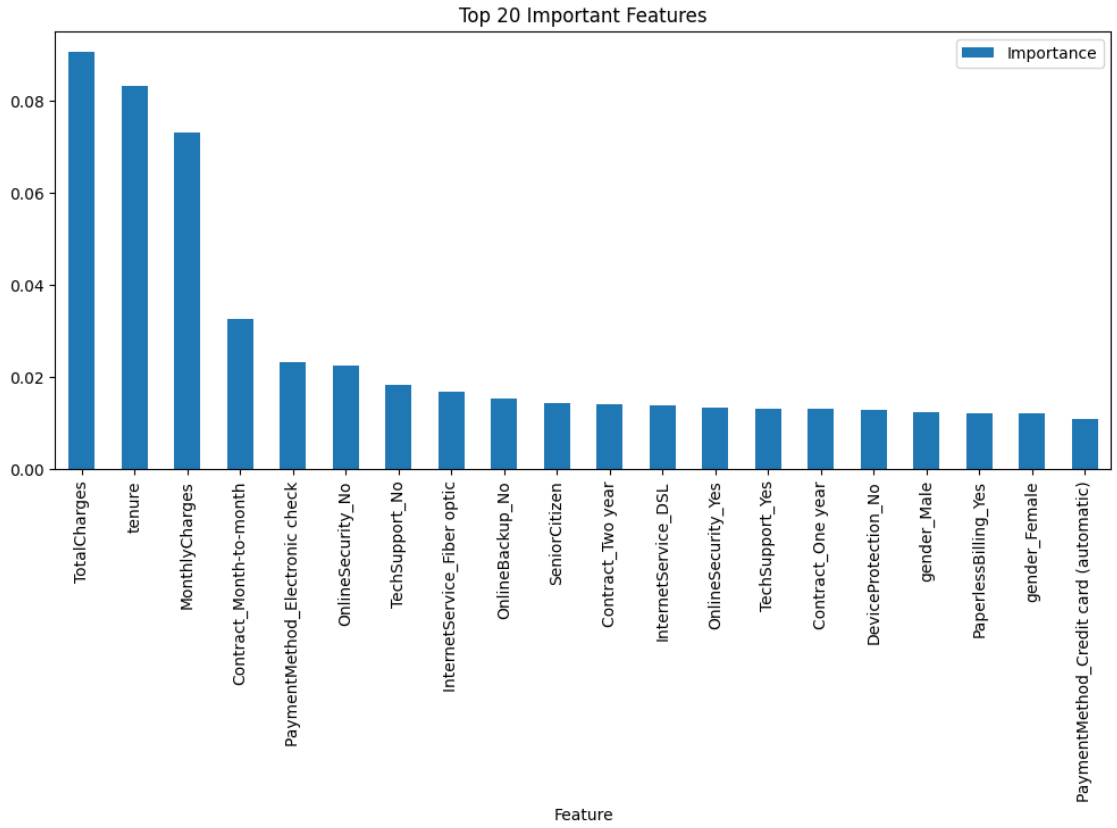
[46]:
```
rf = model.named_steps["classifier"]
ohe = model.named_steps["preprocess"].named_transformers_["cat"]
encoded_cat_cols = ohe.get_feature_names_out(categorical_cols)

all_features = np.concatenate([encoded_cat_cols, numeric_cols])

importances = rf.feature_importances_

imp_df = pd.DataFrame({
    "Feature": all_features,
    "Importance": importances
}).sort_values(by="Importance", ascending=False)

imp_df.head(20)
imp_df.head(20).plot(kind='bar', x='Feature', y='Importance', figsize=(12,5))
plt.title("Top 20 Important Features")
plt.show()
```



[47]:
```
joblib.dump(model, "churn_model.pkl")
```

[47]:
```
['churn_model.pkl']
```