codsoft-ml-2

August 5, 2024

TASK-2 CUSTOMER CHURN PREDICTION

```
[1]: import pandas as pd
     from sklearn.model_selection import train_test_split, GridSearchCV
     from sklearn.preprocessing import StandardScaler, OneHotEncoder
     from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
     from sklearn.linear_model import LogisticRegression
     from sklearn.metrics import classification_report, confusion_matrix,␣
      ↪accuracy_score
```

```
[2]: #Load data
     df = pd.read_csv('/content/WA_Fn-UseC_-Telco-Customer-Churn.csv')
```

```
[3]: df.head()
```

```
[3]:    customerID  gender  SeniorCitizen Partner Dependents  tenure PhoneService  \
     0  7590-VHVEG  Female              0     Yes         No       1           No
     1  5575-GNVDE    Male              0      No         No      34          Yes
     2  3668-QPYBK    Male              0      No         No       2          Yes
     3  7795-CFOCW    Male              0      No         No      45           No
     4  9237-HQITU  Female              0      No         No       2          Yes

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

       TechSupport StreamingTV StreamingMovies        Contract PaperlessBilling  \
     0          No          No              No  Month-to-month              Yes
     1          No          No              No        One year               No
     2          No          No              No  Month-to-month              Yes
     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]
```

[4]:
```
print(df.columns)
```

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

[5]:
```
df.shape
```

[5]:
```
(7043, 21)
```

[6]:
```
# Data preprocessing
# Encode categorical variables with One-Hot Encoding
df.replace(' ', pd.NA, inplace=True)
df.dropna(inplace=True)

cat_cols = ['gender', 'Partner', 'Dependents', 'PhoneService',
            'MultipleLines', 'InternetService', 'OnlineSecurity',
            'OnlineBackup', 'DeviceProtection', 'TechSupport',
            'StreamingTV', 'StreamingMovies', 'Contract',
            'PaperlessBilling', 'PaymentMethod']


df_encoded = pd.get_dummies(df, columns=cat_cols, drop_first=True)
```

[7]:
```
# Split data into features (X) and target (y)
X = df_encoded.drop(['customerID', 'Churn'], axis=1)  # Assuming 'Churn' is the␣
 ↪target variable
y = df_encoded['Churn']
```

[8]:
```
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=28)
```

[9]:
```
# Standardize features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

[10]:
```
# Model 1: Logistic Regression
log_reg = LogisticRegression(max_iter=1000, random_state=28)
log_reg.fit(X_train, y_train)

# Predictions and Evaluation
y_pred_log_reg = log_reg.predict(X_test)
print("Logistic Regression:")
print(classification_report(y_test, y_pred_log_reg))
print(confusion_matrix(y_test, y_pred_log_reg))
print(accuracy_score(y_test, y_pred_log_reg))
```

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

          No       0.86      0.90      0.88      1020
         Yes       0.70      0.60      0.65       387

    accuracy                           0.82      1407
   macro avg       0.78      0.75      0.76      1407
weighted avg       0.81      0.82      0.82      1407

[[920 100]
 [154 233]]
0.8194740582800284
```

[11]:
```
# Model 2: Random Forest Classifier
rf = RandomForestClassifier(n_estimators=50, random_state=28)
rf.fit(X_train, y_train)
```

[11]:
```
RandomForestClassifier(n_estimators=50, random_state=28)
```

[12]:
```
# Predictions and Evaluation
y_pred_rf = rf.predict(X_test)
print("\nRandom Forest Classifier:")
print(confusion_matrix(y_test, y_pred_rf))
print(classification_report(y_test, y_pred_rf))
print("Accuracy:", accuracy_score(y_test, y_pred_rf))
```

```

Random Forest Classifier:
[[930  90]
 [200 187]]
              precision    recall  f1-score   support

          No       0.82      0.91      0.87      1020
         Yes       0.68      0.48      0.56       387

    accuracy                           0.79      1407
```

```
   macro avg       0.75      0.70      0.71      1407
weighted avg       0.78      0.79      0.78      1407

Accuracy: 0.7938877043354655
```