03_baseline_model

March 9, 2024

1 Model

[8]:
```
import sys
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import roc_auc_score
from imblearn.over_sampling import SMOTE
from imblearn.pipeline import Pipeline as ImbPipeline  # To use with SMOTE
```

2 Load Clean Data

[9]:
```
training_set_features = pd.read_csv('../data/clean/training_set_features.csv',␣
 ↪index_col="respondent_id")
training_set_labels = pd.read_csv('../data/clean/training_set_labels.csv',␣
 ↪index_col="respondent_id")
test_set_features = pd.read_csv('../data/clean/test_set_features.csv',␣
 ↪index_col="respondent_id")
```

3 Model Data

[10]:
```
# Separate features and targets
features = training_set_features
h1n1_target = training_set_labels['h1n1_vaccine']
seasonal_target = training_set_labels['seasonal_vaccine']

# Identifying all features as categorical (including binary)
categorical_features = features.columns

# Preprocessing for categorical data including binary
categorical_transformer = Pipeline(steps=[
    ('onehot', OneHotEncoder(handle_unknown='ignore'))  # One-hot encoding for␣
 ↪all features
```

```
])

# Preprocessor for all features
preprocessor = ColumnTransformer(
    transformers=[
        ('cat', categorical_transformer, categorical_features)
    ])

# Building the pipeline for h1n1 vaccine prediction with preprocessing and␣
 ↪class imbalance handling
pipeline_h1n1 = ImbPipeline(steps=[
    ('preprocessor', preprocessor),
    ('oversample', SMOTE(sampling_strategy='auto', random_state=42)),  # Handle␣
 ↪class imbalance
    ('classifier', LogisticRegression(random_state=42, max_iter=1000))  #␣
 ↪Logistic regression classifier
])

# Building the pipeline for seasonal vaccine prediction with preprocessing
pipeline_seasonal = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression(random_state=42, max_iter=1000))  #␣
 ↪Logistic regression classifier
])

# Splitting the dataset into training and validation sets
X_train, X_val, y_train_h1n1, y_val_h1n1, y_train_seasonal, y_val_seasonal =␣
 ↪train_test_split(
    features, h1n1_target, seasonal_target, test_size=0.2, random_state=42)

# Fitting the models
pipeline_h1n1.fit(X_train, y_train_h1n1)
pipeline_seasonal.fit(X_train, y_train_seasonal)

# Making predictions (probabilities) on the validation set
probabilities_h1n1 = pipeline_h1n1.predict_proba(X_val)[:, 1]  # Probabilities␣
 ↪for the positive class
probabilities_seasonal = pipeline_seasonal.predict_proba(X_val)[:, 1]  #␣
 ↪Probabilities for the positive class

# Evaluating the models using ROC-AUC
roc_auc_h1n1 = roc_auc_score(y_val_h1n1, probabilities_h1n1)
roc_auc_seasonal = roc_auc_score(y_val_seasonal, probabilities_seasonal)

print(f"ROC-AUC for H1N1 Vaccine Prediction: {roc_auc_h1n1}")
print(f"ROC-AUC for Seasonal Vaccine Prediction: {roc_auc_seasonal}")
```

```
ROC-AUC for H1N1 Vaccine Prediction: 0.8206701880005715
ROC-AUC for Seasonal Vaccine Prediction: 0.8556704842798477
```

4 Save Model

[11]:
```python
import joblib

# Specify the path to save the model
model_path_h1n1 = "../models/pipeline_h1n1.joblib"
model_path_seasonal = "../models/pipeline_seasonal.joblib"

# Save the models
joblib.dump(pipeline_h1n1, model_path_h1n1)
joblib.dump(pipeline_seasonal, model_path_seasonal)

print(f"Model saved to {model_path_h1n1}")
print(f"Model saved to {model_path_seasonal}")
```

```
Model saved to ../models/pipeline_h1n1.joblib
Model saved to ../models/pipeline_seasonal.joblib
```

5 Submission with Probabilities

[12]:
```python
# Assuming your models are already trained on the training set

# Making predictions on the test set
probabilities_h1n1 = pipeline_h1n1.predict_proba(test_set_features)[:, 1]  # Probabilities for h1n1_vaccine
probabilities_seasonal = pipeline_seasonal.predict_proba(test_set_features)[:, 1]  # Probabilities for seasonal_vaccine

# Creating a DataFrame with the probabilities
submission_df = pd.DataFrame({
    "respondent_id": test_set_features.index,  # Using the index from the test set
    "h1n1_vaccine": probabilities_h1n1,
    "seasonal_vaccine": probabilities_seasonal
})

# Saving the DataFrame to a CSV file
submission_file_path = "../submissions/submission.csv"
submission_df.to_csv(submission_file_path, index=False)

print(f"Submission file with {submission_df.shape[0]} rows saved to {submission_file_path}")
```

```
Submission file with 26708 rows saved to ../submissions/submission.csv
```

6 Submission

Submitted with score: 0.8229 with rank #1359

Top ranked #1 score: 0.8658 AUROC