SI 670 Applied Machine Learning, Homework 3

Please export this notebook as pdf with file name `si670f25_hw3_youruniqname.pdf` when submitting on gradescope.

As a reminder, the work you submit must be your own work. Feel free to discuss general approaches to the homework with classmates: if you end up forming more of a team discussion on multiple questions, please include the names of the people you worked with at the top of your notebook file.

Your name:

Your uniqname:

Setup

In this assignment you will train several linear classifier models and evaluate how effectively they predict instances of fraud using data based on this dataset from Kaggle. Then you'll perform a grid search to find optimal parameters.

Each row in `fraud_data.csv` corresponds to a credit card transaction. Features include confidential variables `V1` through `V28` as well as `Amount` which is the amount of the transaction.

The target is stored in the `class` column, where a value of 1 corresponds to an instance of fraud and 0 corresponds to an instance of not fraud.

In [1]:
```
# # run this cell if you are using Jupyter
files = {'fraud_data.csv': 'fraud_data.csv'}
```

In [2]:
```
import warnings
warnings.filterwarnings("ignore")

%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split

from sklearn.model_selection import train_test_split

df = pd.read_csv(files['fraud_data.csv'])

X = df.iloc[:,:-1]
y = df.iloc[:,-1]
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
```

Question 1 (10 points)

Using `X_train`, `X_test`, `y_train`, and `y_test` (as defined above), train a dummy classifier that classifies everything as the majority class of the training data. What is the accuracy of this classifier? What is the recall?

Then train a LogisticRegression classifier with C=1. What is the accuracy? What is the recall?

This function should a return a tuple with four floats, i.e. `(dummy_accuracy, dummy_recall, lr_accuracy, lr_recall)`.

In [ ]:
```
from sklearn.dummy import DummyClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, recall_score
from sklearn.preprocessing import StandardScaler

def answer_one():
    # YOUR CODE HERE

    dummy = DummyClassifier(strategy = 'most_frequent')
    dummy_accuracy = # YOUR CODE HERE
    dummy_recall = # YOUR CODE HERE

    lr = LogisticRegression(C=1)

    lr_accuracy = # YOUR CODE HERE
    lr_recall = # YOUR CODE HERE

    return dummy_accuracy, dummy_recall, lr_accuracy, lr_recall

answer_one()
```

Question 2 (10 points)

Fit the LogisticRegression with `C` varying from `[[0.1, 1, 10]` and report the accuracy, precision, recall, and F1 scores for each choice of `C`.

This function should a return a tuple with four lists, i.e. `(accuracy_list, precision_list, recall_list, f1_list)`*, and each list should contain 3 numbers.*

In [ ]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler

def answer_two():

    # Accuracy = TP + TN / (TP + TN + FP + FN)
    # Precision = TP / (TP + FP)
    # Recall = TP / (TP + FN)  Also known as sensitivity, or True Positive Rate
    # F1 = 2 * Precision * Recall / (Precision + Recall)

    accuracy_list = []
    precision_list = []
    recall_list = []
    f1_list = []
    for this_C in [0.1, 1, 10]:
        lr = LogisticRegression(C=this_C)
        # YOUR CODE HERE
    return accuracy_list, precision_list, recall_list, f1_list

answer_two()
```

Question 3 (10 points)

In class, we saw how using time of the day as is as a feature creates trouble for us because of a sharp discontinuity. Our solution was to use the sine and cosine functions to create two new features. Briefly explain why using only sine or only cosine would not be sufficient as a solution?

Your answer to Question 3 here

Question 4 (20 points)

You are evaluating some TikTok videos to see if they express pro-Harris or pro-Trump attitudes during the presidential election. Given the confusion matrix below, please

(1) just look at the confusion matrix without calculating, summarize the main problem with this classifier.

(2) first compute the precision, recall, FNR, FPR, F1 score and accuracy for each class, then compute the micro-average of these metrics.

Actual label	Predicted Pro-Trump	Predicted Pro-Harris	Predicted Neutral
Pro-Trump	30	80	33
Pro-Harris	10	70	29
Neutral	52	27	49

Your answer to Question 4 here

Question 5 (10 points)

Examine the code snippet provided below. Identify all the data leakage in it, and explain how to fix the code.

```
In [ ]: from sklearn.datasets import load_breast_cancer
        from sklearn.preprocessing import StandardScaler
        from sklearn.linear_model import LogisticRegression

        X_cancer, y_cancer = load_breast_cancer(return_X_y=True)

        scaler = StandardScaler()
        X_cancer_scaled = scaler.fit_transform(X_cancer)

        X_train, X_test, y_train, y_test = train_test_split(X_cancer_scaled, y_cancer, rand

        lr = LogisticRegression().fit(X_train, y_train)
        print('Breast cancer dataset')
        print('Logistic regression accuracy score: {:.3f}'.format(lr.score(X_test, y_test))
```

Your answer to Question 5 here

Normalizes all of the data at the same time in line 8. Instead, data should be split into training and test first, and then normalized separately. Also, the scaler should only be fit to training data.

Question 6 (10 points)

You are trying to build a new and effective ML model for Michigan Medicine that can detect breast cancer from patient data. You evaluate your new model using accuracy as a metric, and find that it achieves good accuracy.

However, your colleague from Michigan Medicine suggests that instead of maximizing overall accuracy, you should be minimizing the false negative rate (FNR). Briefly explain why neither of these criteria are a good idea to optimize against. Name two criteria that might work better.

Your answer to Question 6 here

Question 7 (20 points)

(1) Suppose that you are trying to solve a binary classification problem (labels 0 or 1), and you have a soft classifier that, for each data point, outputs the probability that the data point has label 1. The table below shows this probability for each data point, along with the true label. Please calculate the respective points on the ROC curve.

Prediction Probability	**0.3**	**0.4**	**0.5**	**0.5**	**0.75**
True label	0	1	0	1	1

(2) Now implement the ROC curve in python code below.

Your answer to Question 7(1) here

```
In [ ]: import matplotlib.pyplot as plt
        from sklearn.metrics import roc_curve, auc

        # True labels and predicted probabilities
        y_true = [0, 1, 0, 1, 1]
        y_scores = [0.3, 0.4, 0.5, 0.5, 0.75]

        # Compute ROC curve and ROC area

        roc_auc = # Your code here

        # Plot ROC curve
        plt.figure(figsize=(6,6))
        plt.plot(fpr, tpr, marker='o', label=f'ROC curve (AUC = {roc_auc:.2f})')
        plt.plot([0, 1], [0, 1], 'k--', label='Random guess')

        plt.xlim([0.0, 1.0])
        plt.ylim([0.0, 1.05])
        plt.xlabel('False Positive Rate')
        plt.ylabel('True Positive Rate')
        plt.title('ROC Curve')
        plt.legend(loc="lower right")
        plt.grid(True)
        plt.show()
```

Question 8 (10 points)

Finish the following code to implement a 5-fold cross validation, and return

(1) the best parameters

(2) the best CV accuracy

(3) the best test precision, recall, and F1

In [ ]:
```python
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Load and split dataset
X, y = load_breast_cancer(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=0, stratify=y
)

# Scale features
scaler = StandardScaler()
# YOUR CODE HERE

# Logistic Regression with GridSearchCV
param_grid = {
    'C': [0.01, 0.1, 1, 10, 100],   # Regularization strength
    'penalty': ['l2'],              # 'l1' requires solver='liblinear' or 'saga'
    'solver': ['liblinear', 'lbfgs']
}

grid_search = # YOUR CODE HERE


print("Best parameters:", # YOUR CODE HERE  )
print("Best CV accuracy:", # YOUR CODE HERE  )
print("Test Accuracy:", # YOUR CODE HERE  )
print("Test Precision:", # YOUR CODE HERE  )
print("Test Recall:", # YOUR CODE HERE  )
print("Test F1:", # YOUR CODE HERE  )
```

Question 9 (30 points)

In this question, you will work with a small dataset of text reviews. Your goal is to build and analyze a sentiment classifier. Complete the following tasks:

Build a pipeline using `CountVectorizer` (or `TfidfVectorizer)` and `LogisticRegression` to classify each review as positive or negative.

Perform hyperparameter tuning with `GridSearchCV`. Explore different settings for both the vectorizer (e.g., n-grams, minimum document frequency, type of vectorizer) and the logistic regression classifier (e.g., regularization strength).

Evaluate your model on a test set. Report performance metrics such as accuracy, precision, recall, and F1-score.

Inspect the top features learned by your model. Identify which words are most strongly associated with positive sentiment and which with negative sentiment.

In [ ]:
```python
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import classification_report
import numpy as np

texts = [
    # Positive
    "I love this product, it's amazing!",
    "Best purchase I’ve ever made.",
    "Really happy with the quality.",
    "Superb item, works perfectly.",
    "Excellent experience, highly recommend.",
    "The service was outstanding and friendly.",
    "Totally worth the money.",
    "Great value and fantastic build.",
    "I enjoy using it every day.",
    "Very satisfied with my order.",

    # Negative
    "Absolutely terrible, I hate it.",
    "Worst experience ever.",
    "I will never buy this again.",
    "Cheap and broke after one use.",
    "Horrible service, very disappointed.",
    "Waste of money, not recommended.",
    "The quality is awful.",
    "Extremely frustrating experience.",
    "It stopped working within a week.",
    "Wouldn’t suggest it to anyone."
]
labels = [
    1,1,1,1,1,1,1,1,1,1,   # positive
    0,0,0,0,0,0,0,0,0,0    # negative
]
```

In [ ]:
```python
# --------------------------------------------------
# Step 1: Train Test Split
# --------------------------------------------------

# TODO: Split into train/test sets (use 70/30 split, random_state=42)
# Hint: use train_test_split
X_train, X_test, y_train, y_test = ...

# --------------------------------------------------
# Step 2: Pipeline
# --------------------------------------------------
# TODO: Define a pipeline that includes:
#   - A vectorizer (either CountVectorizer or TfidfVectorizer)
#   - A LogisticRegression classifier
# Use step names "vect" and "logreg".
pipe = Pipeline([
    ...
])
```

```python
# ---------------------------------------------------
# Step 3: Parameter Grid
# ---------------------------------------------------
# TODO: Create a parameter grid for GridSearchCV that explores:
#   - CountVectorizer vs. TfidfVectorizer
#   - ngram_range = (1,1) and (1,2)
#   - min_df = 1 and 2
#   - LogisticRegression C = [0.01, 0.1, 1, 10]
#   - LogisticRegression penalty = 'l2'
#   - LogisticRegression solver = 'lbfgs'
param_grid = [
    {
        ...
    }
]

# ---------------------------------------------------
# Step 4: Grid Search
# ---------------------------------------------------
# TODO: Initialize a GridSearchCV with:
#   - estimator = pipe
#   - param_grid = param_grid
#   - cv = 3
#   - scoring = "f1"
#   - n_jobs = -1
grid = ...

# TODO: Fit the grid search on the training data
...

# TODO: Print the best parameters and CV score
print("Best parameters:", ...)
print("Best CV score:", ...)

# ---------------------------------------------------
# Step 5: Evaluation
# ---------------------------------------------------
# TODO: Predict on the test set and print classification report
y_pred = ...
print("\nTest set performance:")
print(classification_report(...))

# ---------------------------------------------------
# Step 6: Feature Inspection
# ---------------------------------------------------
# TODO: Extract the vectorizer and classifier from the best model
#       Then print the top 10 positive and top 10 negative features.
best_model = ...
vectorizer = ...
clf = ...

feature_names = ...
coefs = ...

top_pos_idx = ...
```

```
top_neg_idx = ...

print("\nTop positive features:")
for idx in ...:
    print(f"{feature_names[idx]}: {coefs[idx]:.3f}")

print("\nTop negative features:")
for idx in ...:
    print(f"{feature_names[idx]}: {coefs[idx]:.3f}")
```

In [ ]: