CSC 4850 / 6850 / DSCI 4850 - Assigment 3 - (400/450 points)

Total points (400 undergraduate / 450 graduate)

Student Name: (write your name here)

Graduate / Undergraduate (select one)

Instructions:

You are to make a copy of this notebook on your own Google Drive (if you don't have one, get one, it is free), and use the exact format provided. Any code needs to go in the code cells, and any 'text' answer/description needs to go in the proper text cell. We will not be looking for answers randomly placed so please read the instructions.

You are to use only the libraries provided in the next code cell. Any additional library is NOT allowed and will cause you to lose all the points that use said library's functions/functionality. You can use any functions given in the class code examples, but be very very careful of lifting anything 'as-is' from the internet as it will be considerered plagiarism.

IMPORTANT: Make sure you use 1234 (for the folds use: 3456, 5678, 7890) for your randomseed/random states. Failure to do so will make your answers not comparable to the answer key and you will get a zero on the whole assignment.

Submission format:

The submission for Assignment two will have two components:

1) You are to create a PDF from the PRINT out of this notebook with all cells executed sequentially. It is the student's responsibility to be able to do this and no excuses will be accepted, no legible PDF = zero grade. So practice and test before submission time. This PDF should be named LastName_FirstName-Assignment3.PDF

2) The student should create a GitHub repository for this assignment and properly title the repository Class_CODE-ClassName-AssignmentTwo. This repository should have a readme file and the Google Colab notebook in it. Note that colab can save a copy directly to GitHub so make sure you test this. Downloading the notebook file and uploading it directly will result in 200 points deduction. The link to your GitHub repository should be included as text/message in the iCollege submission drop, failure to include this link will result in a 100 point penalty.

Extra Credit for all:

Any student can get 20 extra credit points by doing one simple thing:

1) Make sure your repo for this assignment has a nice README file with figures and results.

```
############## These are the only imports allowed to solve this homework, so make sure you do not add anything else down below

%matplotlib inline

import numpy as np
import pandas as pd
import sklearn
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
```

Using the Chess dataset from Kaggle: https://www.kaggle.com/datasets/datasnaek/chess

```
### Manually download it and upload to this istance data sample space
### Note DO NOT change these operations or all your answers will be incorrect

### Let's do some transformations and extra features on this.
df=pd.read_csv('games.csv', encoding='utf-8')

# Difference between white rating and black rating - independent variable
df['rating_difference']=df['white_rating']-df['black_rating']

# White wins flag (1=win vs. 0=not-win) - dependent (target) variable
df['white_win']=df['winner'].apply(lambda x: 1 if x=='white' else 0)
```

For this assignment we will be using two columns as features only, and the white_win colum as the label.

```
X=df[['rating_difference', 'turns']]
y=df['white_win'].values
```

Question 1 (10 points)

Use sklearn to split this the data into testing and training data.

```
##### Code block for Question 1
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.35, train_size=0.65, random_state=1234)
```

Question 2 (30 points)

Manually (DO NOT use kFold or any built-in functionality) create **THREE** different folds for the training data.

```
##### Code block for Question 2
lengthX = int(len(X_train)/3)
lengthy = int(len(y_train)/3)
foldsX = []
foldsy = []
for i in range(2):
    foldsX += [X_train[i*lengthX:(i+1)*lengthX]]
foldsX += [X_train[2*lengthX:len(X_train)]]
for i in range(2):
    foldsy += [y_train[i*lengthy:(i+1)*lengthy]]
foldsy += [y_train[2*lengthy:len(y_train)]]
```

Question 3 (30 points)

Create code to build three different SVM models with the following kernels:

1. linear
2. poly
3. rbf

```
##### Code block for Question 3
from sklearn.svm import SVC
kernel = ['linear', 'poly', 'rbf']
for i in kernel:
  model = SVC(kernel = i, C = 1.0)
```

Question 4 (70 points)

FOLD 1 - run the first three models with first fold data you created. Output the classification report AND plot its learning curve.

In the text cell, following the code block, descibe what findings can be infered from the classification report and learning curve. Mention at least 3 non-trivial observations between the different kernels.

```
##### Code block for Question 4
from sklearn.model_selection import learning_curve
from sklearn.metrics import classification_report
for i in kernel:
  model = SVC(kernel = i)
  model.fit(foldsX[0], foldsy[0])
  y_predict = model.predict(foldsX[0])
  print(classification_report(foldsy[0], y_predict))
  train_sizes, train_scores, test_scores = learning_curve(model, foldsX[0], foldsy[0], cv = 10, n_jobs = -1,train_sizes = np.linspace(0.01, 1
  train_mean = np.mean(train_scores, axis = 1)
  train_std = np.std(train_scores, axis = 1)
  test_mean = np.mean(test_scores, axis = 1)
  test_std = np.std(test_scores, axis = 1)
  fig, p = plt.subplots(3, 1)
  p[i].plot(train_sizes, train_mean)
```

```
p[i].plot(train_sizes, test_mean)
```

```
              precision    recall  f1-score   support

           0       0.64      0.67      0.66      2145
           1       0.67      0.63      0.65      2200

    accuracy                           0.65      4345
   macro avg       0.65      0.65      0.65      4345
weighted avg       0.65      0.65      0.65      4345

[learning_curve] Training set sizes: [  39  469  899 1329 1759 2189 2619 3049 3479 3910]
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
/usr/local/lib/python3.9/dist-packages/joblib/parallel.py in retrieve(self)
    937                 if getattr(self._backend, 'supports_timeout', False):
--> 938                     self._output.extend(job.get(timeout=self.timeout))
    939                 else:

 ---------------------------------------- 12 frames ----------------------------------------
KeyboardInterrupt:

During handling of the above exception, another exception occurred:

KeyboardInterrupt                         Traceback (most recent call last)
/usr/lib/python3.9/threading.py in _wait_for_tstate_lock(self, block, timeout)
   1078
   1079         try:
-> 1080             if lock.acquire(block, timeout):
   1081                 lock.release()
   1082                 self._stop()

KeyboardInterrupt:
```

SEARCH STACK OVERFLOW

The second kernel has significantly less accuracy than the other two. The first kernel has the highest accuracy with the most samples on both the training set and the test set. The third kernel is best described as the average of the first two.

Question 5 (70 points)

FOLD 2 - run the first three models with first fold data you created. Output the classification report AND plot its learning curve.

In the text cell, following the code block, descibe what findings can be infered from the classification report and learning curve. Mention at least 3 non-trivial observations between the different kernels.

```
##### Code block for Question 5
for i in kernel:
  model = SVC(kernel = i)
  model.fit(foldsX[1], foldsy[1])
  y_predict = model.predict(foldsX[1])
  print(classification_report(foldsy[1], y_predict))
  train_sizes, train_scores, test_scores = learning_curve(model, foldsX[1], foldsy[1], cv = 10, scoring='accuracy', n_jobs = -1, train_sizes
  train_mean = np.mean(train_scores, axis = 1)
  train_std = np.std(train_scores, axis = 1)
  test_mean = np.mean(test_scores, axis = 1)
  test_std = np.std(test_scores, axis = 1)
  fig, p = plt.subplots(3, 1)
  p[i].plot(train_sizes, train_mean)
  p[i].plot(train_sizes, test_mean)
```

The second kernel has significantly less accuracy than the other two. The first kernel seems to have the lowest accuracy on the test set and the highest accuracy on the training set.

Question 6 (70 points)

FOLD 3 - run the first three models with first fold data you created. Output the classification report AND plot its learning curve.

In the text cell, following the code block, descibe what findings can be infered from the classification report and learning curve. Mention at least 3 non-trivial observations between the different kernels.

```
##### Code block for Question 6
for i in kernel:
  model = SVC(kernel = i)
  model.fit(foldsX[2], foldsy[2])
  y_predict = model.predict(foldsX[2])
  print(classification_report(foldsy[2], y_predict))
  train_sizes, train_scores, test_scores = learning_curve(model, foldsX[2], foldsy[2], cv = 10, scoring='accuracy', n_jobs = -1, train_sizes
  train_mean = np.mean(train_scores, axis = 1)
  train_std = np.std(train_scores, axis = 1)
  test_mean = np.mean(test_scores, axis = 1)
  test_std = np.std(test_scores, axis = 1)
  fig, p = plt.subplots(3, 1)
  p[i].plot(train_sizes, train_mean)
  p[i].plot(train_sizes, test_mean)
```

The second kernel has the highest accuracy on this fold. The third kernel has the lowest accuracy on the training set. The first kernel has the lowest accuracy on the test set.

Question 7 (30 points)

From the three folds pick the best model for each different type of kernel.

Present a table with the following columns from their metrics and model. Remember to make classifications on the test set at this stage.

1. Model Name (Kernel)
2. Accuracy
3. Precision
4. Recall
5. F1-score
6. RMSE

```
##### Code block for Question 7
```

Question 8 (40 points)

From question 7, which one is the best model in the following contexts:

a) Metrics from table from question 7, and why?

b) Based on the learning curves ploted in the previous questions, and why?

Textual answer to question 8a goes here.

Textual answer to question 8b goes here.

Question 9 (50 points)

Wrie the simplest and most efficient Sklearn pipeline to do extactly what we did in questions 2 to 6. Make sure that you get all the same intermediate outputs and output the same table from quetsion 7 directly from this pipeline.

```
##### Code block for Question 9
```

Graduate Student Question: (50 points)

Use the following function and provide visualizations for the best models for each kernel type from above (looking for three plots to receive full credit). Note: The function might need some small adjustments :)

```
def Plot_3D(X, X_test, y_test, clf):

    # Specify a size of the mesh to be used
    mesh_size = 5
    margin = 1

    # Create a mesh grid on which we will run our model
    x_min, x_max = X.iloc[:, 0].fillna(X.mean()).min() - margin, X.iloc[:, 0].fillna(X.mean()).max() + margin
    y_min, y_max = X.iloc[:, 1].fillna(X.mean()).min() - margin, X.iloc[:, 1].fillna(X.mean()).max() + margin
    xrange = np.arange(x_min, x_max, mesh_size)
    yrange = np.arange(y_min, y_max, mesh_size)
    xx, yy = np.meshgrid(xrange, yrange)

    # Calculate predictions on grid
    Z = clf.predict_proba(np.c_[xx.ravel(), yy.ravel()])[:, 1]
    Z = Z.reshape(xx.shape)

    # Create a 3D scatter plot with predictions
    fig = px.scatter_3d(x=X_test['rating_difference'], y=X_test['turns'], z=y_test,
                    opacity=0.8, color_discrete_sequence=['black'])

    # Set figure title and colors
    fig.update_layout(#title_text="Scatter 3D Plot with SVM Prediction Surface",
                      paper_bgcolor = 'white',
                      scene = dict(xaxis=dict(backgroundcolor='white',
                                              color='black',
                                              gridcolor='#f0f0f0'),
                                   yaxis=dict(backgroundcolor='white',
                                              color='black',
                                              gridcolor='#f0f0f0'
                                              ),
                                   zaxis=dict(backgroundcolor='lightgrey',
                                              color='black',
                                              gridcolor='#f0f0f0',
                                              )))
    # Update marker size
    fig.update_traces(marker=dict(size=1))

    # Add prediction plane
    fig.add_traces(go.Surface(x=xrange, y=yrange, z=Z, name='SVM Prediction',
                              colorscale='RdBu', showscale=False,
                              contours = {"z": {"show": True, "start": 0.2, "end": 0.8, "size": 0.05}}))
    fig.show()


##### Code block for Graduate Student Question - Figure 1


##### Code block for Graduate Student Question - Figure 2


##### Code block for Graduate Student Question - Figure 3
```