Problem of the Week 6

Name:

Student ID:

Date:

Motivation

This week's Problem of the Week is provides a reflection on the ensemble learning technique we encountered this week in lecture 6-1 - the majority voting approach.

Completing this assignment

To complete this assignment, fill in the markdown cells with your responses. When complete, upload this notebook as a Jupyter notebook AND a PDF to Canvas.

```
# import any python modules you need for this assignment here
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.linear_model import Perceptron, LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import VotingClassifier, BaggingClassifier,
RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve, roc_auc_score
```

Problem 6.1: Penguins Revisited

In homework 2, you implemented two different classifiers to assign sexes to penguins based on morphometric features. In the coding cells below, read in the penguin data as you did for homework 2 and re-implement your classifiers.

```
# Read in the penguin data and standardize it
df = pd.read_csv('adelie_penguins.csv')

df['Classification'] = 1
df.loc[df['Sex'] == 'M', 'Classification'] = 0
df = df.dropna()

df['depth_norm'] = (df['Bill_depth'] - np.mean(df['Bill_depth'])) /
np.std(df['Bill_depth'])
df['length_norm'] = (df['Bill_length'] - np.mean(df['Bill_length'])) /
np.std(df['Bill_length'])
```

```
depth_norm = df['depth_norm']
length_norm = df['length_norm']

x = np.column_stack([depth_norm, length_norm])
X = np.column_stack([df['Bill_depth'], df['Bill_length']])
y = df['Classification']

X_train, X_test, y_train, y_test = train_test_split(x, y, test_size =
0.3, random_state = 17)

# Implement your first classifier
model1 = LogisticRegression()
model1.fit(X_train, y_train)

y_pred = model1.predict(X_test)
y_pred_train = model1.predict(X_train)

print("training loss:", np.mean(y_pred_train != y_train))
print("test loss:", np.mean(y_pred != y_test) )
```

```
training loss: 0.22
test loss: 0.24503311258278146
```

```
# Implement your second classifier
model2 = KNeighborsClassifier(n_neighbors=3)
model2.fit(X_train, y_train)

y_pred = model2.predict(X_test)
y_pred_train = model2.predict(X_train)

print("training loss:", np.mean(y_pred_train != y_train))
print("test loss:", np.mean(y_pred != y_test))
```

```
training loss: 0.1457142857142857
test loss: 0.2582781456953642
```

Problem 6.2: Majority Voting

Next, leverage your two classifiers to implement a majority voting classifier.

```
# implement a majority voting classifier
voters = [('LogisticRegression', LogisticRegression()),
          ('KNN', KNeighborsClassifier(n_neighbors = 3))]
voting_clf = VotingClassifier(voters)

voting_clf.fit(X_train, y_train)
y_pred_vote = voting_clf.predict(X_test)

y_pred = voting_clf.predict(X_test)
y_pred_train = voting_clf.predict(X_train)
```

```
print("training loss:", np.mean(y_pred_train != y_train))
print("test loss:", np.mean(y_pred != y_test))
```

```
training loss: 0.17142857142857143
test loss: 0.2781456953642384
```

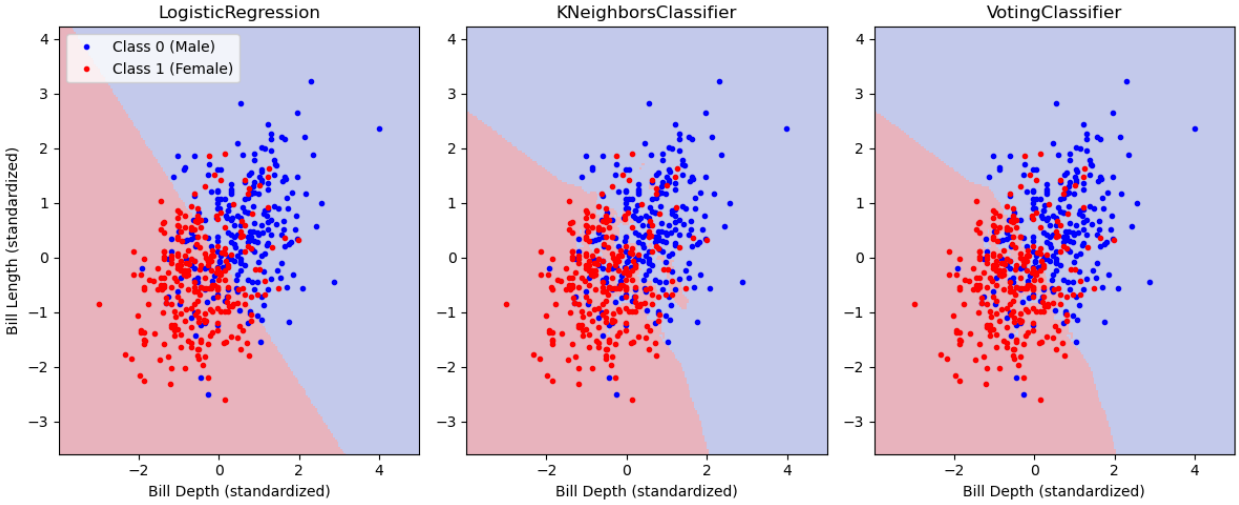
Problem 6.3: Classification Plots

In class, we wrote a `classify_domain` function to shade our plots according to the classification. Implement that here and create three side-by-side plots with the results of each of your classifiers (the two from homework 2, and the new majority voting classifier).

```
# create your plot here
def classify_domain(model, X_data, y):
    X1, X2 = np.meshgrid(np.linspace(x_min, x_max, 200),
                         np.linspace(y_min, y_max, 200))

    Y_pred = model.predict(np.column_stack([X1.ravel(),
X2.ravel()])).reshape(X1.shape)

    return X1, X2, Y_pred, x_min, x_max, y_min, y_max


fig = plt.figure(figsize=(12,5))

models = [LogisticRegression(),
          KNeighborsClassifier(n_neighbors=3),
          VotingClassifier(voters)]

fig = plt.figure(figsize=(12,5))

for m, model in enumerate(models):
    model.fit(X_train, y_train)

    X1, X2, Y_pred, x_min, x_max, y_min, y_max =
classify_domain(model, x, y)

    plt.subplot(1, 3, m + 1)
    plt.pcolormesh(X1, X2, Y_pred, cmap='coolwarm', alpha=0.3,
shading='auto')
    plt.plot(x[y==0,0], x[y==0,1], 'b.', label='Class 0 (Male)')
    plt.plot(x[y==1,0], x[y==1,1], 'r.', label='Class 1 (Female)')

    plt.title(model.__class__.__name__)
    plt.gca().set_xlim([x_min, x_max])
    plt.gca().set_ylim([y_min, y_max])
    plt.xlabel("Bill Depth (standardized)")

    if m == 0:
```

```
        plt.ylabel("Bill Length (standardized)")
        plt.legend(loc=2)

plt.tight_layout()
plt.show()
```

```
<Figure size 1200x500 with 0 Axes>
```



Problem 6.4: Model Interpretation

How is your new majority voting model performing relative to the previous classifiers? How do you know whether your model is better or worse? Write 3-5 sentences below answer these questions. Be sure to include any pertinent plots or metrics that have supported your conclusions.

It's hard to say whether it's strictly better or worse. Visually it's not overfitting the way KNN is with some of the little clusters here and there creating little "islands" of female classification, and it has more nuance than the strict logistic regression model. However, after calculating the loss, it seems to fit worse than both. I think this is because it attempts to create a compromise between the moddels, it misses out on the mathematical optimization of the logistic regression and the little "islands" of KNN. It is a more well-rounded model but it loses out on the precise strengths of the models it uses and thus is actually weaker in the end.