Homework 2

Instructions

For undergraduates, please answer problems 1 through 5. Graduate students answer all of the questions. If a question allows for you to show your work, it is recommended you show that work because partial credit will be rewarded. Please submit as a pdf. For parts of the homework you do on paper, scan it in such that it is legible (there are a number of free Android/iOS scanning apps, if you do not have access to a scanner). This assignment is due on Gradescope on **09/10** at **23:59 EDT**.

This homework (and many subsequent ones) will involve data analysis and reporting on methods and results using Python code. You have to submit a single PDF file that contains everything to Gradescope, and associated each page of the PDF to each problem. This includes any text you wish to include to describe your results, the complete code snippets of how you attempted each problem, any figures that were generated, and scans of any work on paper that you wish to include. It is important that you include enough detail that we know how you solved the problem, since otherwise we will be unable to grade it.

I recommend that you use Jupyter/iPython notebooks to write your report. It will help you not only ensure all of the code for the solutions is included, but also provide an easy way to export your results to a PDF file 1. I recommend liberal use of Markdown cells to create headers for each problem and sub-problem, explaining your implementation/answers, and including any mathematical equations. For parts of the homework you do on paper, scan it in such that it is legible (there are a number of free Android/iOS scanning apps, if you do not have access to a scanner), and include it as an image in the iPython notebook2. If you have any questions/concerns about using iPython, ask us on Piazza. If you decide not to use iPython notebooks, but go with Microsoft Word or Latex to create your PDF file, you have to make sure all of the answers can be generated from the code snippets included in the document.

Summary so far: (1) submit a single, standalone PDF report, with all code; (2) I recommend Jupyter notebooks.

Points: This homework adds up to a total at **40** points. Each of problems 1 through 5 are worth 8 points for undergraduates and 7 points for graduate students. Problem 6 is worth 2 points for graduate students and problem 7 is worth 3 points for graduate students.

Problem 1: Data Preprocessing

Load the Titanic dataset from OpenML with the following code:

```python
from sklearn.datasets import fetch_openml

titanic = fetch_openml(data_id=40945, as_frame=True)

```

```
X = titanic.data
Y = titanic.target
```

Answer the following questions.

1. Use the `shape` method to find the shape of the feature data. Report the numbers and report what each of the numbers mean.

2. Use the `isnull` method to count the number `NaN` values in the feature data. Report the numbers per feature. Use the `dropna` method to drop all of the features with more than 300 `NaN` values. Report the numbers of features remaining. Count the number of remaining `NaN` values and drop the remaining samples with `NaN` values. Report the number of samples remaining.

3. Use the `drop` method to drop the `name` and `ticket` columns. Use the `pandas.get_dummies` function to convert the categorical data to dummy variables. Report the number of features after the transformations.

4. Use `describe` method to provide the summary statistics of the data. Report the mean and standard deviation for the `age` and `fare` columns.

5. For the `age` and `fare` features, provide a scatter plot where the color indicates survival or not.

Problem 2: Naive Bayes

Part A

Consider a small dataset of passengers with the following features and survival outcomes:

Passenger	Sex	Class	Survived
1	Male	1st	No
2	Female	2nd	Yes
3	Male	3rd	No
4	Female	1st	Yes
5	Female	2nd	No

Note that each passenger is considered to be equally likely, i.e. $\Pr(Passenger1) = \Pr(Passenger2) = \cdots = 0.2$. Using this data:

1. Calculate the prior probabilities $\Pr(Survived)$ and $\Pr(NotSurvived)$.

2. Compute the likelihood probabilities $\Pr(Sex|Survived)$ and $\Pr(Class|Survived)$ for all possible values.

3. Use the naive Bayes formula to calculate the probability of survival for a new passenger who is a female traveling in 2nd class. Show all steps of your calculation.

Part B

Convert the data and targets from problem 1 to `numpy.ndarrays` and split the data using the following code. If you used a different variable for the cleaned `X`, be sure to use that variable name here:

```python
from sklearn.model_selection import train_test_split

# Align the targets with the features
drop_indices = list(set(Y.index.to_list()) - set(X.index.to_list()))
Y = Y.drop(drop_indices)

# Pull the binary features
columns = ['sex_female', 'sex_male', 'embarked_C', 'embarked_Q', 'embarked_S']
X_nb = X[columns]

# Convert to numpy
X_nb, Y_nb = X_nb.to_numpy(), Y.to_numpy()

# Train, test split
X_nb_tr, X_nb_te, Y_nb_tr, Y_nb_te = train_test_split(X_nb, Y_nb, test_size=0.2, random_state=123)
```

Answer the following questions.

4. Import, initialize, and fit the `BernoulliNB` classifier on the `X_nb_tr` and `Y_nb_tr`. Report the accuracy, auc, and log loss on the training set.

5. Compute the posterior predictions and predictive probability on the test set. Report the accuracy, auc, and log loss on the test set. How do these metrics compare to the training metrics?

Problem 3: k-Nearest Neighbors

Returning to your data frame, use the following code to select just the features `fare` and `age`:

```python
# Pull the two features
columns = ['age', 'fare']
X_nn = X[columns]

# Convert to numpy
X_nn, Y_nn = X_nn.to_numpy(), Y.to_numpy()
```

Answer the following questions.

1. Complete the `predict` and `predict_proba` methods for the `KNN` class in the `ml_tools.py` file by computing the Euclidean distance between the training data and inputs.

2. Use the `MinMaxScaler` from `sklearn` to rescale the data `X_nn` in the range $[-1, 1]$.

3. Split the data using `train_test_split` with 0.2 test size with a random state of 123. Fit the model on the training data with `k=1` and to the number of samples in your training data using the `KNN` class from the `ml_tools`. Then evaluate the performance of your fitted model. Report accuracy, auc, and log loss on the training and holdout test set. Do the same using `sklearn`'s `KNeighborsClassifier`.

4. Use `DecisionBoundaryDisplay.from_estimator` from `sklearn` to plot the decision boundary of the `KNeighborsClassifier` classifier on your data.

Problem 4: Linear Regression

Load the diabetes dataset from `sklearn` using the following code.

```python
from sklearn.datasets import load_diabetes

diabetes = load_diabetes()

X_db = diabetes.data
Y_db = diabetes.target
```

Answer the following questions.

1. Complete the `fit` method in the `ml_tools.py` for the `LinearRegression` class by computing the Ordinary Least Squares.

2. Use the `StandardScaler` from `sklearn` to rescale the data `X_db`. Report the mean and the standard deviation of the rescaled data using `numpy`'s `mean` and `std`.

3. Split the data using `train_test_split` with 0.2 test size with a random state of 123. Fit a `LinearRegression` class model from `ml_tools` using training data. Then evaluate the performance on the mean squared error and report the error on training and test set. Fit several `Ridge` class from `sklearn` with alpha's $[0.1, 0.25, 0.5, 1.0]$. Compute the performance on mean squared error for all models on training and test set. Report all errors. Which model performed best and why?

Problem 5: Logistic Regression

Use the cleaned data `X` from problem 1 and the aligned targets `Y` from problem 2 for this problem. Answer the following questions.

1. Complete the `sigmoid` and `update_coeff` methods for the `LogRegression` class in the `ml_tools.py` file by computing the gradients. Hint: you can

compute the partial derivatives by taking the matrix product of the derivative of the model output with respect to the cross-entropy loss, which is $\hat{y} - y$, and the transpose of the derivative of the coefficients with respect to the inputs, which is the input matrix.

2. Use the `StandardScaler` from `sklearn` to rescale the continous and integer data in `X`. Report the mean and the standard deviation of the `fare` feature.

3. Split the data using `train_test_split` with 0.2 test size with a random state of 123. Do 10-fold cross validation using `LogRegression` class from `ml_tools` and the `LogisticRegression` class from `sklearn` with the penalty set to `l2`. Compute the accuracy, auc, and log loss for each fold. Report the average accuracy, auc, and log loss for each model.

4. Fit both models on the full training data and plot the ROC curve on hold-out test set using `RocCurveDisplay` for both models. How does the model performance on the hold-out test set compare to the cross-validation results?

Problem 6: Bernoulli Naive Bayes (Graduate Students)

Explain why we use a `BernoulliNB` naive Bayes classifier in problem 2. Can we use this same classifier on the full dataset `X`? Why or why not.

Problem 7: Advanced Logistic Regression (Graduate Students)

Modify the `fit` method for the `LogRegression` class in `ml_tools.py` to compute the average training log loss on each step of the epoch and to compute an average test log loss on the hold-out test set. Store the respective average losses and return them in a list. Use the modified `fit` method on the full training data and hold-out test data. Plot the results using `matplotlib.pyplot`.