Human-Centered Machine Learning
ISE 5194, Spring 2021
Professor Krening

Homework #2

Problem 1:

A supervised learning model was trained to classify songs by genre. For example, a 10 second audio file of the song 'Georgia On My Mind' by Ray Charles is given as input to the classifier, which outputs the genre of 'jazz'.

a. A 10 second audio file of Op. 55 No. 1 by Chopin was classified by the algorithm as belonging to the 'rock' genre. This was an incorrect classification. What is this incorrect classification called?
    a. This type of incorrect classification is called loss.
b. Is it possible to create a classification algorithm that never makes this kind of mistake? Why or why not?
    a. It's not possible to have a "no loss" algorithm. This Is because machine learning algorithms are built on induction, and at large scales there will always be instance where the algorithm hasn't come across a particular or outlier data point and will make an incorrect classification.
c. With a closer analysis of the situation, a researcher has determined that the classification algorithm did make a mistake, but its mistake made sense given the training data used to build the model. The classifier made the best decision it could given the information it had at the time. How do we describe this situation?
    a. This situation would be described as no regret, where the algorithm made the best decision it could given its current information.

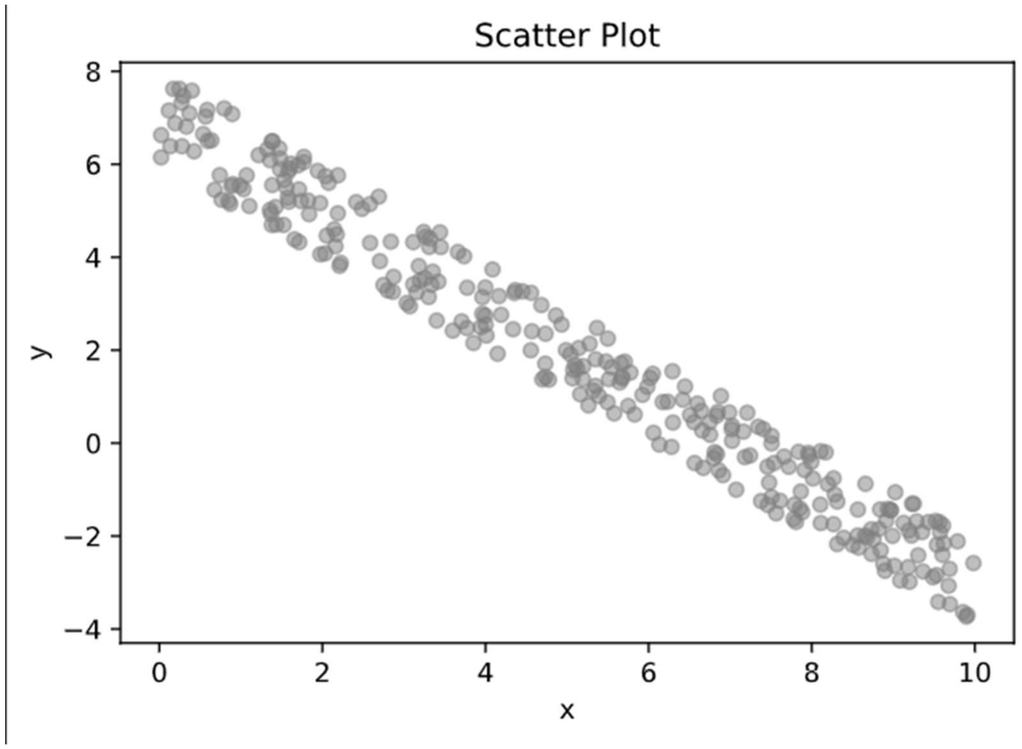
Problem 2: Regression

Your task is to implement a linear regression algorithm for a single-input single-output system from scratch. A SISO linear model is defined as $y_i = w_0 + w_1 x_i$, where w_0 is the y-intercept, w_1 is the slope of the line, x_i is a scalar input, and y_i is the scalar output value. You will use the following files in this problem:

File in the *hw3/problem1/* directory	Description
linearRegressionSingleFeature_prob2.py	Template code. You will add your code to this file. It contains: • *main*: function that loads data and calls a plotting function. You will need to add some things to this function.

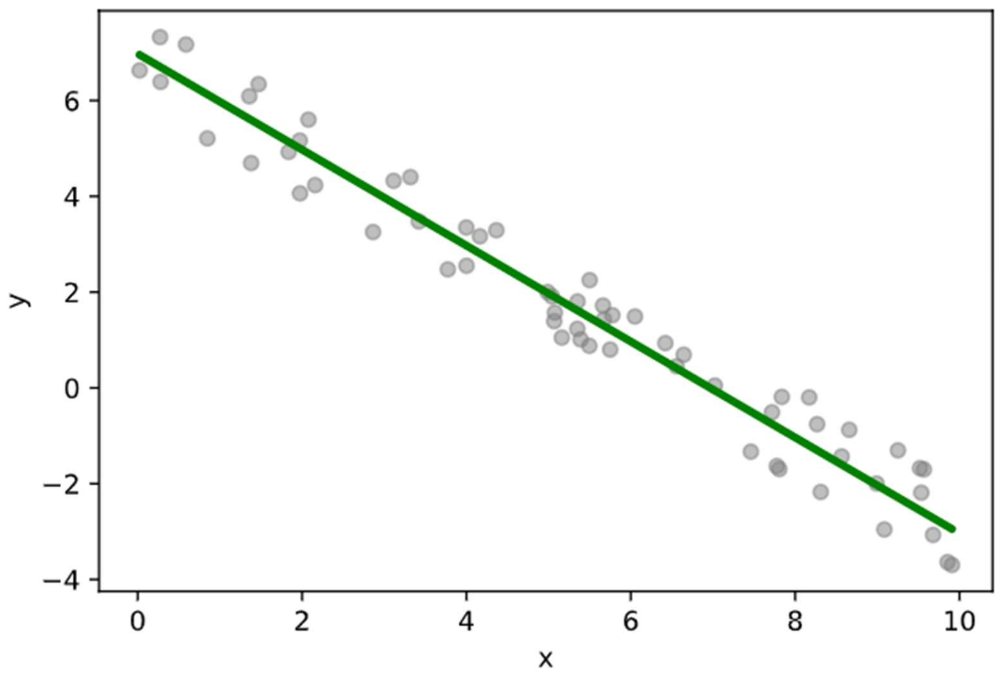
	• *plot_regression_line*: function to make plots. This is for your benefit. Modify or change it as much as you wish. • *estimate_coeffs*: function outline to estimate the coefficients for linear regression.
Homework2_linearRegression_data.csv	Input data.

i. Make a scatter plot showing the input data.



ii. Do you think we should use linear regression on this data? Why or why not?
    a. Yes I think linear regression should be used on this data. Looking at the scatter plot, there seems to be a strong linear correlation between x and y, and it looks like there's a best fit line that could represent the data well.
iii. Split your data into training and testing sets. Explain how you accomplished this. Why do we split data like this? You are allowed to use existing software for this step.
    a. The data was split into training and testing data using the train_test_split function from sklearn. This is a clean way to randomly split the data
iv. In the *estimate_coeffs* function, write code to calculate the model coefficients.
v. For the provided data, use your *estimate_coeffs* function to fit a linear regression model. What are the calculated values of the linear regression model coefficients?
    a. The slope coefficient came to be -1.001, and the y intercept was determined to be 6.976.
vi. Given the input data, calculate the predicted output.

vii. Make a scatter plot showing the data and the model.



viii. How accurate is your model? You are allowed to use existing software for this step.

a. The R^2 value for this model is 0.9608

ix. Copy your code here.

```
# Assignment 3, Problem 1
# Purpose:  Build a linear regression model that predicts output
#           based on a single input feature.


import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression
from sklearn import metrics


#
# Function:   estimate_coeffs
# Purpose:    To calculate the intercept and slope of the
#             regression line based on the training data
# Input:
```

```
#     x:      input training data
#     y:      training output
# Output:
#     ( w0, w1): intercept and slope of the linear regression model
#
def estimate_coeffs(x, y):

    # reshape data as .fit() wants 2D arrays, and currently x and y are only 1D a
rrays
    x = x.reshape(-1,1)
    y = y.reshape(-1,1)

    x_mean = np.mean(x)
    y_mean = np.mean(y)
    numerator = 0
    denominator = 0

    for i in range(len(x)):
        numerator += (x[i] - x_mean)*(y[i] - y_mean)
        denominator += (x[i] - x_mean)*(x[i] - x_mean)

    slope = numerator / denominator
    intercept = y_mean - slope*x_mean


    return ( intercept, slope )


#
# Function:   plot_regression_line
# Purpose:    To plot the x and y data along with the
#             trained linear model.
# Input:
#     x:      input training data
#     y:      training output
#     w:      [0] intercept and [1] slope of trained model

#    NOTE: Actually plotting testing data not training data
#
def plot_regression_line(x, y, w):
    # plot data points
    plt.scatter(x, y, color = "gray", alpha=0.5, marker = "o", s = 30)

    # calculate prediction vector
    y_pred = w[0] + w[1]*x
    # plot the regression line
    plt.plot(x, y_pred, color = "g", linewidth = 3 )

    plt.xlabel('x')
    plt.ylabel('y')

    plt.show()


def main():
    # Import data.
    npData = np.array(data)
    y = npData[:,0]
    x = npData[:,1]


    #################################################
    x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.2)
    #################################################


    #################################################
    # To Do: Call the estimate_coeffs function to calculate
    # the weights for the 'best fit' line.
    w = estimate_coeffs(x_train, y_train)
    #################################################


    print ( "Estimated Coefficients:" )
    print ( "  w0: " + np.str(w[0]) + "   w1: " + np.str(w[1]) )

    # Plot the regression line with data
    plot_regression_line ( x_test, y_test, w )


if __name__ == "__main__":
    main()
```

Code Block to make scatter plot of points

```
npData = np.array(data)
plt.scatter(npData[:,1], npData[:,0], color = "gray", alpha=0.5, marker = "o", s
= 30)
plt.xlabel('x')
plt.ylabel('y')
plt.title('Scatter Plot')
```

Code block to get the accuracy of the linear regression model

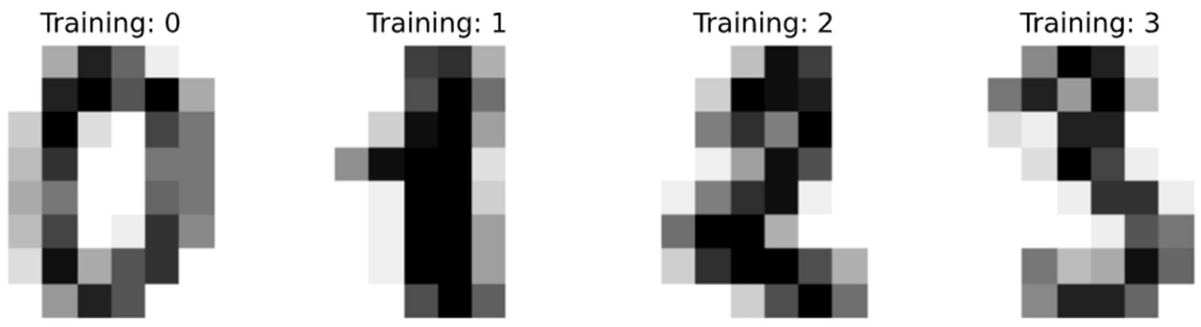
```
npData = np.array(data)
y = npData[:,0]
x = npData[:,1]
x = x.reshape(-1,1)
y = y.reshape(-1,1)
reg = LinearRegression()
reg.fit(x, y)
print("The R^2 of this model is: ")
print(str((reg.score(x, y)).round(4)))
```

Note: You may use existing implementations of linear regression (such as from sci-kit learn) to *test* your code, but you must write your own linear regression code.

Problem 3: Classification

Your task is to classify images containing handwritten digits into their corresponding integer number (0-9). For this problem, use the functions provided in the python sci-kit learn package.

i. Import the handwritten digits dataset from the sci-kit learn package. https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html
ii. Plot the first four images.



iii. Before training a model, is there anything you need to do to this data? Why or why not?
    a. Before training the model, we first need to flatten the images to a 1D array, as rectangular “shapes” can’t be direct inputs.
iv. Train a Random Forest Classifier to predict the digit based on an input image.
    a. What is the model’s accuracy?
        i. The accuracy of the random forest classifier is 96.67%

b. Copy your code here.

```
# declare, train, and test random forest classifier
rfc = RandomForestClassifier()
rfc.fit(x_train, y_train)
predict = rfc.predict(x_test)
print("The accuracy of this model is: ")
print(str((rfc.score(x_test, y_test)*100).round(2)) + "%")
```

v. Train a Neural Network Classifier to predict the digit based on an input image.
   a. What is the model's accuracy?
      i. The accuracy of this model is 96.11%
   b. Copy your code here.

```
# declare, train, and test neural network classifier
nnc = MLPClassifier()
nnc.fit(x_train, y_train)
print("The accuracy of this model is: ")
print(str((nnc.score(x_test, y_test)*100).round(2)) + "%")
```

vi. Use Recursive Feature Elimination to train a Random Forest Classifier on the most important 25 features.
   a. List the top 5 features (by pixel location).
      i. The top 5 features are: (3,6), (4,6), (6,5), (2,5), and (5,6)
   b. What is the model's accuracy?
      i. The accuracy of this model is 96.39%
   c. Copy your code here.

```
# declare random forest classifier using 25 most important features in recursive
feature selection
estimator = RandomForestClassifier()
rfe = RFE(estimator=RandomForestClassifier(), n_features_to_select=25)
fit = rfe.fit(x_train, y_train)
# ranking = rfe.ranking_.reshape(digits.images[0].shape)
ranking = fit.ranking_
print(ranking)
print("The accuracy of this model is: ")
print(str((rfe.score(x_test, y_test)*100).round(2)) + "%")
```

vii. Use Recursive Feature Elimination to train a Random Forest Classifier on the most important 8 features.

a. What is the model’s accuracy?
    i. The accuracy of this model is 92.50%

b. Copy your code here.

```
# declare random forest classifier using 8 most important features in recursive f
eature selection
estimator = RandomForestClassifier()
rfe = RFE(estimator=RandomForestClassifier(), n_features_to_select=8)
rfe.fit(x_train, y_train)

print("The accuracy of this model is: ")
print(str((rfe.score(x_test, y_test)*100).round(2)) + "%")
```

viii. List potential sources of error in this problem.
  a. The data could have misclassified labels.
  b. The model could be overfitting with the training data.
  c. There could be some cases of data that are noisy or difficult to interpret.
  d. The data could have bias. For example, there could be twice as many zeros as there are twos, so the model would learn more into getting zeros correct at the possible cost of getting twos correct. Though looking at the source data it seems that the labels are uniformly distributed.

ix. Are decision trees explainable? Why or why not?
  a. Decision trees are explainable. The final decision can be represented visually, where humans are able to look at where the decision tree decided to split off, and can go into the algorithm and determine why the algorithm determined to split on a particular feature at that particular node.

x. Are random forests explainable? Why or why not?
  a. Random forests may be able to be interpretated at smaller scales, but by in large, for any large set of data random forests are not explainable. Though each tree can be explained, the culmination of all the trees and their unique intricacies makes it impractical to be easily interpreted by a human.

xi. Are neural networks explainable? Why or why not?
  a. Neural networks are not explainable. For large problems, there may be multiple hidden layers, each with their own distinctive weights and inputs. A visual representation of a robust neural network would look really convoluted are very difficult for a human to understand all the ins and outs.

Problem 4:

i. Provide an example of how supervised learning could be applied to the medical field.
  a. One possible application is using imagining data to classify between benign and cancerous tumors. The imaging scans can be divided into training and testing data, and a classifier such as a decision tree or neutral net could be used to determine what class the image is.

ii. What is overfitting?

        a. Overfitting is when a model builds itself too closely to a set of data. Though this makes the training accuracy high, a model that tries to exactly reflect the training data will likely have trouble with new data as it's not as generalized.

iii. What is a method to prevent overfitting?
    a. One method to prevent overfitting is node pruning, which is used for decision tree classifiers. Node pruning will cut or prevent the splitting of nodes that don't add much value, seem to be very specific to the training data, or are redundant.

iv. What does generalization mean in machine learning?
    a. Generalization in ML is a model's ability to adapt to new unseen data. A common example is holding some data from a data set as testing data, which is data that the model did not use to build itself.

v. Imagine you are paying someone to label a dataset of images with natural language captions describing what is going on in the image. What are potential issues? Could these issues impact the model you train? Why or why not?
    a. Potential issues are:
        i. You and the other person may disagree on the classification.
        ii. If there aren't set rules, the person may not be consistent in their classification.
        iii. The person may make a mistake in mislabeling an image.
    b. These issues would definitely impact the model. Human error is obvious, as the model will likely diverge from the ideal model using incorrect data. Similar issues will come up with the potential for inconsistent classification. Finally, disagreement between you and the other person may make it harder for you to interpret your model and data.

Extra Credit Problem 1:

List at least 5 potential ways human input can be incorporated into a supervised learning algorithm.

1. Physical human data can be used in a machine learning algorithm. Data such as heart rate, height, body weight, etc can be inputs for a model.
2. Non physical human data can be used. The Likert scale is one way to turn human thoughts and preferences as data for a model.
3. Humans can help classify data labels for a machine learning model. A classic example is websites asking humans to determine which pictures have traffic lights and which pictures do not. Here humans are helping to label image data.

4. Humans can help in the testing / validation of supervised learning models. By correcting the model, the algorithm can learn from its mistakes and improve in each generation.
5. This one is a bit specific, but humans are an important in types of problems such as NLP. One example is if a model uses Naïve Bayes to classify emails as spam or not spam, it can find common words and their counts in each email as data. Though for language in particular, the ordering of words is very important, and humans can help models determine if the ordering of words is correct or not.