CMPT 120 Project: Machine Learning for Prediction

Machine learning systems are all around us, e.g. understanding images, speech recognition, medical diagnosis, automated stock trading, prediction systems, etc. The heart of any machine learning system is building a model that is exposed to a large number of inputs and the applicable output for them. Once the model has analysed more and more data, it tries to figure out the relationship between input and the result.

As a simple example, let's say we need to decide whether to wear a jacket or not depending on the temperature. The *training data* for this problem might look like:

Outside Temperature	Wear a Jacket
30 °C	No
25 °C	No
20 °C	No
15 °C	Yes
10 °C	Yes

Looking at this, we can think of a connection between the input (temperature) and the output (decision to wear a jacket). For example, if the temperature is 12 °C, you would still wear a jacket even though you were never told the outcome for that particular temperature. That's called making a prediction!

Getting Started

To build our machine learning model, we need to use the *scikit-learn* package:

To work online:

- Repl.it has the *scikit-learn* package available already; we just need to import it as shown in the examples that follow

To work on your own machine, choose one of the following options:

- Install *scikit-learn* using *pip* on your system, after installing Python/IDLE from python.org:
    - https://scikit-learn.org/stable/install.html

Building our first Machine Learning Model

Step 1: Figuring out the algebraic relationship between the inputs and the output

Consider the following set of inputs (**X**), and output **y**:

X			
x_1	x_2	x_3	y
1	2	3	14
4	5	6	32
11	12	13	74
21	22	23	134
5	5	5	30

Can you see an algebraic relationship between the set of inputs **X** and the output **y**? Hint: think of some integer coefficients c_1, c_2, and c_3 that would solve the equation:

$$y = c_1 \times x_1 + c_2 \times x_2 + c_3 \times x_3$$

Try different rows of values in the table above for x_1, x_2, x_3, and y in the above equation.

To check your answer, go to the main Project page in Canvas:
https://canvas.sfu.ca/courses/56176/pages/final-project-machine-learning-for-prediction

Step 2: Generating a training set

Write a Python program that creates two lists:

- The list of inputs will be a list of lists, where each item in the list is a list of 3 values [x1, x2, x3]
    - The input list will have 100 three-value sublists inside it
    - Each sublist will consist of 3 random integers between 0 and 1000
    - Use the **randint()** function from the **random** module to generate the random integers
- The list of outputs will have 100 items, where each item is computed using the algebraic relationship you discovered in Step 1 applied to one sublist of the input list

For example, the first few entries in the list of inputs might look like:

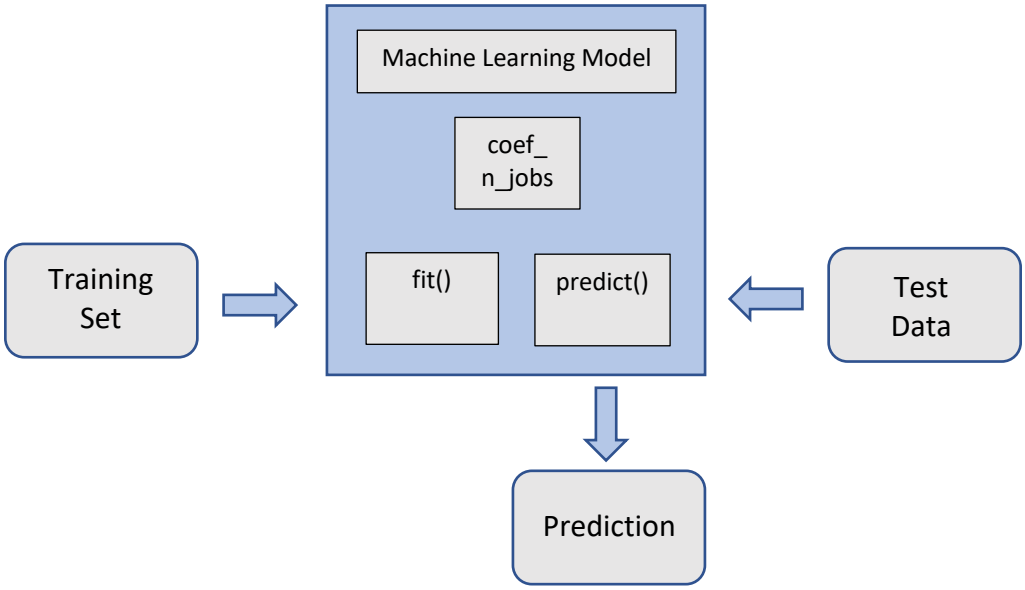
[[837, 700, 970], [350, 935, 610], [544, 161, 203], [78, 625, 396], [649, 794, 340], [119, 937, 881], [206, 422, 563], …]

And the corresponding first few entries in the list of outputs would look like:

[5147, 4050, 1475, 2516, 3257, 4636, 2739, …]

Step 3: Training the machine learning model

The *scikit-learn* package gives us an easy way to build a machine learning model. It allows us to train the model using the training set we've generated, and then use the model to predict an output for new input (the *test data*). To train the model, we feed our training set to the fit() function, which will train and create the model. To use the model to predict an output, we feed our test data to the predict() function, which will generate the predictions using the model.



We have the training set ready, so we can create a model and pass it our training data as below. This assumes our training set (the one you created in Step 2) is made up of the training input in a list called *train_input*, and the training output in a list called *train_output*.

```
from sklearn.linear_model import LinearRegression

predictor = LinearRegression(n_jobs=-1)
predictor.fit(X=train_input, y=train_output)
```

Step 4: Using our model for prediction

We can use our model to predict what the outcome should be for the new test input [10, 20, 30]:

```
X_test = [[10, 20, 30]]
outcome = predictor.predict(X=X_test)
coefficients = predictor.coef_

print("Prediction: " + str(outcome))
print("Coefficients: " + str(coefficients))
```

Do the prediction and coefficients match with what you expected to get using your algebraic formula?

Building a Model Using Real Data

Download the *SeoulBikeData.csv* file from:
https://archive.ics.uci.edu/ml/datasets/Seoul+Bike+Sharing+Demand

The file contains bike sharing data for the city of Seoul, Korea from 2017-2018. We would like to predict the count of rented bikes based on all the other numeric values in the .csv file. Carefully read the description of the dataset given at the link above.

Step 1: Process the data

You will need to do the following before starting:

- Check to see if there is a header row in the file
- Make sure the file (especially the header) doesn't contain any strange characters like ° (degrees)
- Identify which column in the file contains the value we would like to eventually predict (i.e., the count of rented bikes)
    - This column is the output for the data fed to our model
- Identify all other numeric columns in the file that can be used to predict the count of rented bikes
    - These columns are the inputs for the data fed to our model

Do the following to process your data in Python:

- Read in all lines of the .csv file, and form two lists:
    - The output list contains all output values
    - The input list is a list of lists, where each item is a list of the numeric values found in one line of the file
    - Make sure all items you place in these lists are floating point values!
- Take the input list and further split it into two lists:
    - One that will be fed to the model as *training input*
    - A second that will be a list of inputs to be used by our finished model for predictions (called the *test input*)

    - Approximately 80% of the overall items in the original input list should be in the training input, and the remaining 20% should be in the test input
- Take the output list and split it into two lists the same way as the input list:
    - One that will be fed to the model as *training output*
    - A second will be a list of outputs to be used to compare the predictions our model produces to the actual values; this list is called the *test output*
    - Approximately 80% of the overall items in the original output list should be in the training output, and the remaining 20% should be in the test output

Step 2: Training your model

Now that we have the *training input* and *training output* lists, we can train and build our model in the same way as previously (see Step 3 from the last section).

Step 3: Using your model for prediction

Use your *test input* list to predict values for the count of rented bikes using your model in the same way done in Step 4 of the last section. This will generate a list of predictions. You can then compare each prediction to the corresponding actual values in the *test output* list to see how good your model's predictions were!

Step 4: Visualizing the performance of your model

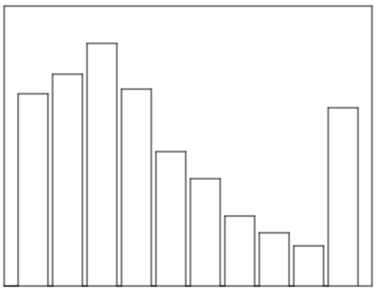
One way to see how your model performed is to calculate how far off each prediction was from the corresponding actual value. To do this for one prediction, we can calculate the *percentage error*, which is the absolute value of the difference between the actual value and the prediction, divided by the actual value. The equation looks like this:

$$percentageError = \frac{abs(actualValue - predictedValue)}{actualValue}$$

Calculate the percentage error for all of your predictions, and count how many of the predictions were within 10% of the actual value, how many were within 10% to 20%, how many were within 20% to 30%, …, up to how many were within 90% to 100%, and finally, how many were more than 100% away from the actual value. Hint: use a dictionary!

Note: you should exclude the calculation of any percentage errors where the actual value is zero, otherwise you will encounter a divide by zero error.

Once you have these counts, you can graph the 10 categories in a diagram using Python with Turtle. For example, your diagram might look something like this:



Notes:

- No numbers or labels are shown in the diagram above. Be creative; your solution should look much better than this!
- If using Python/IDLE on your own system: you can use Python with Turtle in the same program that uses *scikit-learn* to build your model. This is the preferred option!
- If using REPL: the *scikit-learn* package is not available in Python with Turtle. You'll have to export the category count data to a .csv file from your regular Python REPL, and then import the .csv file to your Python with Turtle REPL.

Now we have a much better idea of how good our model's predictions are!

Project Extensions

Here are some ways to extend your project:

- Get a dataset from below, understand it, process the data, train a model, predict, and visualize your results as above
    - Facebook performance metrics data set: https://archive.ics.uci.edu/ml/datasets/Facebook+metrics
    - Stock portfolio performance data set: https://archive.ics.uci.edu/ml/datasets/Stock+portfolio+performance
    - Forest Fires data set: https://archive.ics.uci.edu/ml/datasets/Forest+Fires
    - Parkinson's Telemonitoring data set: https://archive.ics.uci.edu/ml/datasets/Parkinsons+Telemonitoring
    - Real Estate Valuation data set: https://archive.ics.uci.edu/ml/datasets/Real+estate+valuation+data+set
    - You can also find a .csv file by searching elsewhere online; just make sure there is a column that can be used as a value to be predicted, and at least a few numeric columns that can be used as inputs
- Build an application that allows the user to provide a .csv file (by placing it in the same folder as your Python program, or by uploading it to your REPL):
    - Your program will ask the user to give the index in the .csv file of the prediction column, and the range of indices for the input columns
    - You will then process the data, train a model, predict, and visualize the results as above