

SMLB Challenge 2 Guide



Students for Machine Learning in Business

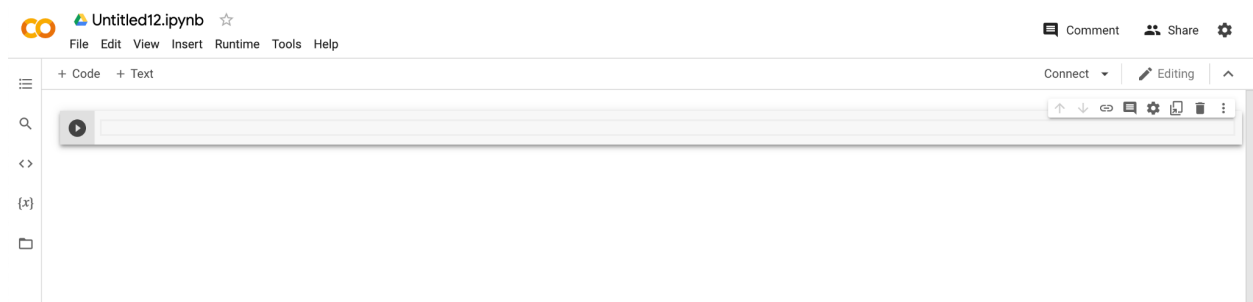
Are you completely new to Python and ML? This document is intended to guide you through the process of creating a working submission using a logistic regression and the scikit-learn Python package. **Please keep in mind that if you're confident in your Python skills, you do not have to follow this walkthrough!** This document is meant to help newcomers get started. There are several ways to do this challenge, this is just one of the many! You don't need to follow this guide. However, please **read section 13!**

Step 1: Load Google Colaboratory

Because your submission needs to be in Python, your first step needs to be finding a developing environment for Python! Google Colab allows you to write jupyter notebooks, a file format that allows for easy Python coding and commenting, while running your programs on Google's computers!

Visit Google Colab at <https://colab.research.google.com/>

Log in with your UAlberta account, and select 'New Notebook.' This notebook is a development environment that allows you to write and run Python programs that you write. Your notebook should look something like this:



Google Colab organizes information in 'cells.' You can click the '+ code' button at the top to add a new coding cell, where you can type python commands and run them by hitting the play button on the side. You can also click the '+ text' button to add a text cell where you can write about any notes along the way. We encourage you to use these text blocks to make your process easily understandable to the markers along the way!

Step 2: Learn a little more about python

Now that you have a way to easily run Python programs, it's time to learn a little more about Python programming. Fortunately, there are so many resources available to learn the basics. Here's one of our favorite videos to get started with:

<https://www.youtube.com/watch?v=rfscVS0vtbw>

While all of the sections of the above video are crucial in understanding Python, if you are low on time, the following video chapters are particularly relevant to this challenge: (This is only needed if you don't know these Python concepts. If you wish to learn as you progress through the challenge, you can skip this step!)

- Variables & Data Types (Starts 15:06, Length: ~13 min)
- Working with Numbers (Starts 38:18, Length: ~10 min)
- List / List Functions (Starts 1:03:10, Length: ~15 min)
- Functions & Returns (Starts 1:24:15, Length: ~15 min)
- Control Flow (if, elif, else) (Starts 1:40:06, Length: ~20 min)
- For Loops (Starts 2:32:44, Length: ~9 min)
- Modules and Pip (Starts 3:28:18, Length ~15 min)

Step 3: Understanding Your Data

It's time to download and understand the data you will need to predict wildfires. All machine learning techniques require a large amount of data to make good predictions, and this challenge is no different.

The data ([download here](#)) you have been given contains columns describing the weather and conditions, date, and whether or not a fire occurs on that day, the next day, or the day after. You can open the data in excel or a similar spreadsheet program that is likely pre-installed on your computer.

Data Legend:

All data is per the given date and forest_area.

Unnamed: 0: Arbitrary index column.

Date: Date of data row.

Forest_area: An ID index for areas in Alberta.

avg_Dry_bulb_temperature: Average air temperature in degrees celsius.

avg_Dew_point: Average dew point in degrees celsius.

avg_Relative_humidity: Average air moisture, expressed as a percentage.

avg_Rain_mm: Average rainfall in millimeters during that day.

avg_Snow_cm: Average snowfall in centimeters during that day.

avg_Hail_mm: Average hail deposition in millimeters during that day.

avg_Precipitation_mm: Sum of rain, snow and hail columns in millimeters.

avg_Wind_speed_kmh: Average wind speed in kilometers per hour.

avg_Wind_gust_kmh: Average gusting wind speed in kilometers per hour.

avg_Wind_azimuth: Average direction of wind in degrees from north. Increasing clockwise.

avg_Ffmc: Represents fuel moisture of forest litter fuels under the shade of a forest canopy.

avg_Dmc: Represents fuel moisture of decomposed organic material underneath the litter.

avg_Dc: Represents drying of fuels deep in the soil.

avg_Isi: A measure of expected rate of fire spread, depends generally on wind speed.

avg_Bui: Represents the amount of fuel available for burning. Depends on DMC and DC.

avg_Fwi: A general fire danger index.

avg_Edr: A fire risk index.

Is_fire: 0 or 1 representing no/yes if a fire occurred on that day in that forest area.

Is_fire_tmrw: 0 or 1 representing no/yes if a fire occurred the next day in that forest area.

Is_fire_2days: 0 or 1 representing no/yes if a fire occurred in two days in that forest area.

While inspecting the data, take notice that some columns contain words and others contain numbers. There are also empty cells or 'NA values', which will need to be dealt with during your regression analysis.

You should also upload the csv data file to your personal Google Drive, so that you can access it directly from your Google Colab.

Step 4: Importing Modules

Reopen Google Colab and start a new project. This will be your submission.

As you learned in Step 2, in order to skip right to implementation, you can import a third-party python module to skip all of the fine details of programming certain types of ML algorithms. You should import the following functions from scikit-learn if you want to do a simple logistic regression (recommended for beginners). Type the following into the first

code cell of your notebook:

```
[ ] #Import the relevant libraries:
import sklearn
import pandas as pd
from sklearn.linear_model import LogisticRegression
import numpy as np
from google.colab import drive
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import plot_roc_curve
```

For your reference, sklearn is [scikit-learn](#), a popular machine learning library. [Pandas](#) is a data science library where you can visualize and preprocess the data. [Numpy](#) is a very common mathematics library, and [google.drive](#) allows you to import the data from your drive into your notebook.

Step 5: Importing Data

First, allow Google Colaboratory access to your files by running the following commands in a new cell:

```
[ ] #import the data:
#run this to grant the notebook access to your google drive
drive.mount('/content/drive/')
```

Log in with your Google account and then load the data into a pandas 'DataFrame' using the following command in a new cell:

```
#Copy path to data file and load it as "raw_dataset"
path = '/content/drive/MyDrive/wildfire_train.csv'
raw_dataset = pd.read_csv(path)
```

Note that the 'path' specified above is the location where you uploaded the training data. You may have to change this if you did not upload the training data to the main home of your google drive.

Here, this line of code reads the .csv file we have stored our training data. It then stores the data into a special object called a pandas DataFrame.

If running `pd.read_csv` is returning an error message, make sure you read the [documentation](#) for the function, or search online for common issues. This is a process that you should get used to, as it is overwhelmingly common that anyone's code will contain at least one bug, (a bug is simply an issue or error with the code). It is recommended that you test your code line by line as it develops so that you can squash any bugs as they appear.

Step 6: Preprocess Data

One of the most important steps in developing machine learning models is fine-tuning the data. In other words, you have to make sure that the data is able to be processed, and is relevant.

In the majority of the cases, machine learning algorithms do not work with strings. You will also have to deal with the blanks in the data, drop any unnecessary columns, split the data into training and validation data, and also scale the data in order to prevent any unintended mathematical errors.

You can start this process by viewing the columns and dropping ones that we will not use for the regression analysis. In this example you will drop any rows that contain empty cells for the greatest simplicity. (This is likely not the best solution).

```
[ ] #Get rid of unnecessary columns that won't be useful for regression as is (categorical variables, etc)

print(raw_dataset.columns) #take a look at the columns


regress = raw_dataset[['avg_dry_bulb_temperature', 'avg_dew_point',
                        'avg_relative_humidity', 'avg_rain_mm', 'avg_snow_cm', 'avg_hail_mm',
                        'avg_precipitation_mm', 'avg_wind_speed_kmh', 'avg_wind_gust_kmh', 'avg_wind_azimuth',
                        'avg_ffmc', 'avg_dmc', 'avg_dc', 'avg_isi', 'avg_bui', 'avg_fwi', 'avg_esr', 'is_fire']] #only include numeric columns

regress = regress.dropna() #get rid of na rows for simplicity
```

Here, 'regress' is a new dataframe that only contains information that we care about for our regression.

Now, we will split our data into a training set and a validation set. The training set is what will be used to teach our model how to classify whether or not a fire will occur on a given day. The validation set is what you will use to make sure your model can make good predictions.


Do this with the following command in a new cell:

```
 #Split into train and validation data  
train_data , validation_data = train_test_split(regress)
```

Now specify what parts of your data will be the training parts to ‘fit’ onto another variable. In other words, ‘X’ will be the data that you use to predict ‘Y.’ In this case we want to use all of the variables to predict ‘is_fire.’

```
[ ] #create x and y data  
  
X = train_data[['avg_dry_bulb_temperature', 'avg_dew_point',  
                'avg_relative_humidity', 'avg_rain_mm', 'avg_snow_cm', 'avg_hail_mm',  
                'avg_precipitation_mm', 'avg_wind_speed_kmh', 'avg_wind_gust_kmh', 'avg_wind_azimuth',  
                'avg_ffmc', 'avg_dmc', 'avg_dc', 'avg_isi', 'avg_bui', 'avg_fwi', 'avg_esr']]  
  
Y = train_data['is_fire']  
  
X_validation = validation_data[['avg_dry_bulb_temperature', 'avg_dew_point',  
                                'avg_relative_humidity', 'avg_rain_mm', 'avg_snow_cm', 'avg_hail_mm',  
                                'avg_precipitation_mm', 'avg_wind_speed_kmh', 'avg_wind_gust_kmh', 'avg_wind_azimuth',  
                                'avg_ffmc', 'avg_dmc', 'avg_dc', 'avg_isi', 'avg_bui', 'avg_fwi', 'avg_esr']]  
  
Y_validation = validation_data['is_fire']
```

Lastly, rescale the data so that no mathematical errors occur during training.

```
 #scale data:  
  
scaler = StandardScaler()  
X = scaler.fit_transform(X)  
X_validation = scaler.transform(X_validation)
```

Step 7: Training Your Simple Model

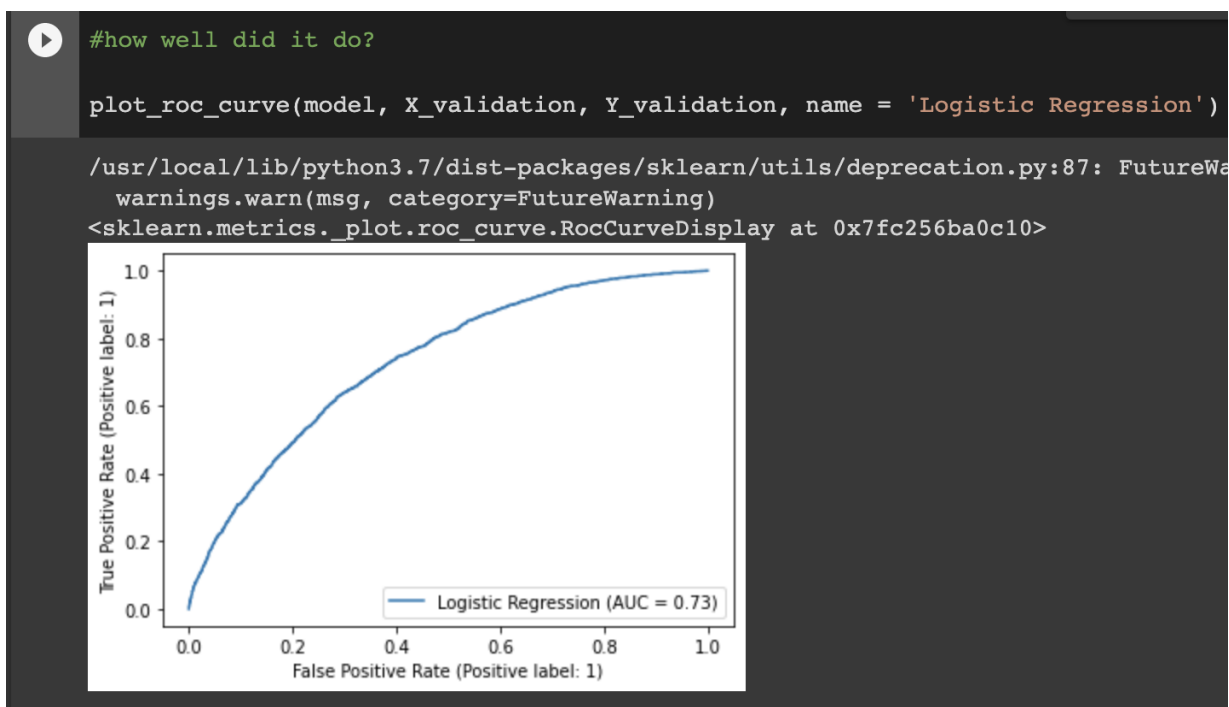
This part demonstrates the beauty of python libraries. All you have to do is type and run this command:

```
[ ] #fit the model:  
  
model = LogisticRegression()  
model.fit(X, Y)
```

And now your model is trained! Surprisingly easy right? Libraries like scikit-learn allow us to skip all of the backend mathematics and just implement a workable solution for our business problem.

Step 8: Test Your Results

When you are developing code, you may want to see how well your model compares with the validation data. Luckily, we set some of that aside earlier! Lets plot an ROC (receiver operating characteristic) curve to visualize the success of our model.



ROC curves are standard ways to visualize the efficacy of your classifier. The [AUC](#) (area under curve) is given in the bottom right corner. Higher values of AUC indicate better performing models. If $AUC = 0.5$, then your classifier is as effective as randomly guessing. If

AUC = 1, your classifier will always select the correct prediction. Your final score will be based on this AUC value.

You are done! You've just trained a machine learning algorithm to predict when wildfires will occur. The rest of this guide will show you how to start improving your model.

Step 10: (OPTIONAL) Optimize Hyperparameters

You should now have a model that can predict same day wildfires with known accuracy. Congratulations! It's now up to you to improve the model to have a shot at winning. There are a multitude of ways to go about this, but we would recommend starting by tuning some of the 'hyperparameters' to figure out what will give you the best results. Hyperparameters are simply details of how your ML algorithm learns, and you probably won't be sure what should be changed until you try it and observe your model's test score going up or down!

If you followed this guide, you will have used the 'LogisticRegression' model from sklearn. To find out details about what hyperparameters you can change about your decision tree, visit:

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html.

This webpage will give you an idea of what inputs the Logistic Regression can take. You should be trying different combinations of inputs to see how your model responds. We will not tell you what you should change as trying things on your own is an extremely important part of coding as a whole. If you are satisfied with the accuracy of your model, you are ready to submit!

Step 11 (OPTIONAL): Predict Fires Tomorrow and the Next Day

Now that you have a model that can predict fires on the same day as the weather reading, see if you can build one for fires the day after or two days after the weather reading! In other words, try to predict the 'is_fire_tmrw' and 'is_fire_2days' columns. If you do not do this, you accept the lowest possible score in these categories.

Step 12 (OPTIONAL): Go Above and Beyond!

We have walked you through the steps of how to implement a very simple logistic regression classifier, but this is by no means the only way to solve this problem. If you are

interested in building a more complex, and potentially more accurate model, you should pay attention to the way the data is structured. Maybe you want to try a model other than a regression. It's possible that a neural network may be able to better predict fires. Make sure you take the time to understand the data. Use online resources and experiment to build the best possible model. The choices are nearly endless. Good luck!

Step 13: How to Submit

When your model is ready to submit, you should try to prepare it so that it is easy for the markers to grade. Your submission should include a cell of commented out code telling the instructors how to upload new data and use it in your model. You need to keep in mind that the test data is formatted exactly like the training data, but excludes the 'is_fire,' 'is_fire_tmrw' and "is_fire_2days' columns. There will be empty cells just as with the training data. If you run empty cells through your regression model it will return an error! To deal with this, we included a command to fill every empty cell with the average value from that column. Try to be as clear as possible with text cells and comments, so that we know how to use your model. Please contact us if you are unsure or run into any issues.

```
[12] #This test data is hidden from competitors, but try to make it easy
      #for the organizaers to import and run the test data through with few alterations.
      #Commenting out the lines of code relevant to the test data would be wise.

      #path = '/content/drive/MyDrive/wildfire_test.csv'

      #test_data = pd.read_csv(path)

      #Get rid of categorical variables so that our model is compatible with the test data
      #test_data = test_data[['avg_dry_bulb_temperature', 'avg_dew_point', 'avg_relative_humidity',

      #Remember, there are na's filled in the test data. One approach to rectifying this is to fill
      #test_data = test_data.fillna(test_data.mean())

      #X_test = test_data[['avg_dry_bulb_temperature', 'avg_dew_point',
                          #'avg_relative_humidity', 'avg_rain_mm', 'avg_snow_cm', 'avg_hail_mm',
                          #'avg_precipitation_mm', 'avg_wind_speed_kmh', 'avg_wind_gust_kmh', 'avg_wind_azimuth',
                          #'avg_ffmc', 'avg_dmc', 'avg_dc', 'avg_isi', 'avg_bui', 'avg_fwi', 'avg_esr']]
      #Y_test = test_data['is_fire']

      #X_test = scaler.transform(X_test)

      #final_predictions = model.predict(X_test)

      #plot_roc_curve(model, X_test, Y_test, name = 'Logistic Regression')
```

Submit the link to your Colaboratory notebook via smlb.org, or click [here](#).

Make sure the 'share' permission settings are adjusted so that

smlb@ualberta.ca has access to view everything.

We will contact you if there are any issues.