Plagiarism Detection Model

Now that you've created training and test data, you are ready to define and train a model. Your goal in this notebook, will be to train a binary classification model that learns to label an answer file as either plagiarized or not, based on the features you provide the model.

This task will be broken down into a few discrete steps:

- Upload your data to S3.
- Define a binary classification model and a training script.
- Train your model and deploy it.
- Evaluate your deployed classifier and answer some questions about your approach.

To complete this notebook, you'll have to complete all given exercises and answer all the questions in this notebook.

> All your tasks will be clearly labeled **EXERCISE** and questions as **QUESTION**.

It will be up to you to explore different classification models and decide on a model that gives you the best performance for this dataset.

Load Data to S3

In the last notebook, you should have created two files: a `training.csv` and `test.csv` file with the features and class labels for the given corpus of plagiarized/non-plagiarized text data.

> The below cells load in some AWS SageMaker libraries and creates a default bucket. After creating this bucket, you can upload your locally stored data to S3.

Save your train and test `.csv` feature files, locally. To do this you can run the second notebook "2_Plagiarism_Feature_Engineering" in SageMaker or you can manually upload your files to this notebook using the upload icon in Jupyter Lab. Then you can upload local files to S3 by using `sagemaker_session.upload_data` and pointing directly to where the training data is saved.

In [1]:

```
import pandas as pd
import boto3
import sagemaker
```

In [2]:

```
"""
DON'T MODIFY ANYTHING IN THIS CELL THAT IS BELOW THIS LINE
"""
# session and role
sagemaker_session = sagemaker.Session()
role = sagemaker.get_execution_role()

# create an S3 bucket
bucket = sagemaker_session.default_bucket()
```

EXERCISE: Upload your training data to S3

Specify the `data_dir` where you've saved your `train.csv` file. Decide on a descriptive `prefix` that defines where your data will be uploaded in the default S3 bucket. Finally, create a pointer to your training data by calling `sagemaker_session.upload_data` and passing in the required parameters. It may help to look at the Session documentation (https://sagemaker.readthedocs.io/en/stable/session.html#sagemaker.session.Session.upload_data) or previous SageMaker code examples.

You are expected to upload your entire directory. Later, the training script will only access the `train.csv` file.

In [3]:

```
# should be the name of directory you created to save your features data
data_dir = 'plagiarism_data'

# set prefix, a descriptive name for a directory
prefix = 'plagiarism-data'

# upload all data to S3
input_data = sagemaker_session.upload_data(path=data_dir, bucket=bucket, key_pre
fix=prefix)
```

Test cell

Test that your data has been successfully uploaded. The below cell prints out the items in your S3 bucket and will throw an error if it is empty. You should see the contents of your `data_dir` and perhaps some checkpoints. If you see any other files listed, then you may have some old model files that you can delete via the S3 console (though, additional files shouldn't affect the performance of model developed in this notebook).

In [4]:

```
"""
DON'T MODIFY ANYTHING IN THIS CELL THAT IS BELOW THIS LINE
"""
# confirm that data is in S3 bucket
empty_check = []
for obj in boto3.resource('s3').Bucket(bucket).objects.all():
    empty_check.append(obj.key)
    print(obj.key)

assert len(empty_check) !=0, 'S3 bucket is empty.'
print('Test passed!')
```

```
plagiarism-data/sagemaker-scikit-lea-200214-2244-001-75a47045/outpu
t/model.tar.gz
plagiarism-data/sagemaker-scikit-lea-200214-2244-002-e8222d91/outpu
t/model.tar.gz
plagiarism-data/sagemaker-scikit-lea-200214-2244-003-6a53fc3a/outpu
t/model.tar.gz
plagiarism-data/sagemaker-scikit-lea-200214-2244-004-ad647036/outpu
t/model.tar.gz
plagiarism-data/sagemaker-scikit-lea-200214-2244-005-e2bbe8a7/outpu
t/model.tar.gz
plagiarism-data/sagemaker-scikit-lea-200214-2244-006-ee30cda8/outpu
t/model.tar.gz
plagiarism-data/sagemaker-scikit-lea-200214-2252-001-9e6044cd/outpu
t/model.tar.gz
plagiarism-data/sagemaker-scikit-lea-200214-2252-002-a22bfea2/outpu
t/model.tar.gz
plagiarism-data/sagemaker-scikit-lea-200214-2252-003-73fa3f16/outpu
t/model.tar.gz
plagiarism-data/sagemaker-scikit-lea-200214-2252-004-9237b32a/outpu
t/model.tar.gz
plagiarism-data/sagemaker-scikit-lea-200214-2252-005-d829b092/outpu
t/model.tar.gz
plagiarism-data/sagemaker-scikit-lea-200214-2252-006-bbcdcd8f/outpu
t/model.tar.gz
plagiarism-data/sagemaker-scikit-lea-200214-2252-007-d558bf06/outpu
t/model.tar.gz
plagiarism-data/sagemaker-scikit-lea-200214-2252-008-d692d39e/outpu
t/model.tar.gz
plagiarism-data/sagemaker-scikit-lea-200214-2252-009-78dfc081/outpu
t/model.tar.gz
plagiarism-data/sagemaker-scikit-lea-200214-2252-010-02d5b3c3/outpu
t/model.tar.gz
plagiarism-data/sagemaker-scikit-lea-200214-2252-011-f2e331f4/outpu
t/model.tar.gz
plagiarism-data/sagemaker-scikit-lea-200214-2252-012-b9ccd474/outpu
t/model.tar.gz
plagiarism-data/sagemaker-scikit-lea-200215-0018-001-4b517ef5/outpu
t/model.tar.gz
plagiarism-data/sagemaker-scikit-lea-200215-0018-002-7d418a49/outpu
t/model.tar.gz
plagiarism-data/sagemaker-scikit-lea-200215-0018-003-39a3f182/outpu
t/model.tar.gz
plagiarism-data/sagemaker-scikit-lea-200215-0018-004-e52dea8b/outpu
t/model.tar.gz
plagiarism-data/sagemaker-scikit-lea-200215-0018-005-d8fc5d50/outpu
t/model.tar.gz
plagiarism-data/sagemaker-scikit-lea-200215-0018-006-8e84b4a5/outpu
t/model.tar.gz
plagiarism-data/sagemaker-scikit-lea-200215-0018-007-f71f39c1/outpu
t/model.tar.gz
plagiarism-data/sagemaker-scikit-lea-200215-0018-008-0e46112b/outpu
t/model.tar.gz
plagiarism-data/sagemaker-scikit-lea-200215-0018-009-f67aaf49/outpu
t/model.tar.gz
plagiarism-data/sagemaker-scikit-lea-200215-0018-010-613a826d/outpu
t/model.tar.gz
plagiarism-data/sagemaker-scikit-learn-2020-02-14-19-05-59-607/debug
-output/training_job_end.ts
plagiarism-data/sagemaker-scikit-learn-2020-02-14-19-13-06-464/debug
-output/training_job_end.ts
plagiarism-data/sagemaker-scikit-learn-2020-02-14-19-13-06-464/outpu
```

```
t/model.tar.gz
plagiarism-data/sagemaker-scikit-learn-2020-02-14-19-24-56-306/debug
-output/training_job_end.ts
plagiarism-data/sagemaker-scikit-learn-2020-02-14-19-29-53-478/debug
-output/training_job_end.ts
plagiarism-data/sagemaker-scikit-learn-2020-02-14-19-36-50-445/debug
-output/training_job_end.ts
plagiarism-data/sagemaker-scikit-learn-2020-02-14-20-05-50-608/debug
-output/training_job_end.ts
plagiarism-data/sagemaker-scikit-learn-2020-02-14-20-05-50-608/outpu
t/model.tar.gz
plagiarism-data/sagemaker-scikit-learn-2020-02-14-20-51-11-400/debug
-output/training_job_end.ts
plagiarism-data/sagemaker-scikit-learn-2020-02-14-20-55-06-513/debug
-output/training_job_end.ts
plagiarism-data/sagemaker-scikit-learn-2020-02-14-20-55-06-513/outpu
t/model.tar.gz
plagiarism-data/sagemaker-scikit-learn-2020-02-14-21-31-14-253/debug
-output/training_job_end.ts
plagiarism-data/sagemaker-scikit-learn-2020-02-14-21-31-14-253/outpu
t/model.tar.gz
plagiarism-data/sagemaker-scikit-learn-2020-02-14-21-42-04-133/debug
-output/training_job_end.ts
plagiarism-data/sagemaker-scikit-learn-2020-02-14-21-42-04-133/outpu
t/model.tar.gz
plagiarism-data/sagemaker-scikit-learn-2020-02-14-22-05-35-719/debug
-output/training_job_end.ts
plagiarism-data/sagemaker-scikit-learn-2020-02-14-22-05-35-719/outpu
t/model.tar.gz
plagiarism-data/sagemaker-scikit-learn-2020-02-14-22-19-55-787/debug
-output/training_job_end.ts
plagiarism-data/sagemaker-scikit-learn-2020-02-14-22-19-55-787/outpu
t/model.tar.gz
plagiarism-data/sagemaker-scikit-learn-2020-02-14-22-37-48-740/debug
-output/training_job_end.ts
plagiarism-data/sagemaker-scikit-learn-2020-02-14-22-37-48-740/outpu
t/model.tar.gz
plagiarism-data/sagemaker-scikit-learn-2020-02-14-23-17-29-681/debug
-output/training_job_end.ts
plagiarism-data/sagemaker-scikit-learn-2020-02-14-23-17-29-681/outpu
t/model.tar.gz
plagiarism-data/sagemaker-scikit-learn-2020-02-14-23-54-36-424/debug
-output/training_job_end.ts
plagiarism-data/sagemaker-scikit-learn-2020-02-14-23-54-36-424/outpu
t/model.tar.gz
plagiarism-data/test.csv
plagiarism-data/train.csv
sagemaker-scikit-learn-2020-02-14-18-36-27-975/source/sourcedir.tar.
gz
sagemaker-scikit-learn-2020-02-14-18-52-52-374/source/sourcedir.tar.
gz
sagemaker-scikit-learn-2020-02-14-19-05-59-607/source/sourcedir.tar.
gz
sagemaker-scikit-learn-2020-02-14-19-13-06-464/source/sourcedir.tar.
gz
sagemaker-scikit-learn-2020-02-14-19-24-56-306/source/sourcedir.tar.
gz
sagemaker-scikit-learn-2020-02-14-19-29-53-478/source/sourcedir.tar.
gz
sagemaker-scikit-learn-2020-02-14-19-36-50-445/source/sourcedir.tar.
gz
```

```
sagemaker-scikit-learn-2020-02-14-20-05-50-608/source/sourcedir.tar.
gz
sagemaker-scikit-learn-2020-02-14-20-51-11-400/source/sourcedir.tar.
gz
sagemaker-scikit-learn-2020-02-14-20-55-06-513/source/sourcedir.tar.
gz
sagemaker-scikit-learn-2020-02-14-21-19-29-030/source/sourcedir.tar.
gz
sagemaker-scikit-learn-2020-02-14-21-22-47-177/source/sourcedir.tar.
gz
sagemaker-scikit-learn-2020-02-14-21-23-37-151/source/sourcedir.tar.
gz
sagemaker-scikit-learn-2020-02-14-21-24-02-753/source/sourcedir.tar.
gz
sagemaker-scikit-learn-2020-02-14-21-31-14-253/source/sourcedir.tar.
gz
sagemaker-scikit-learn-2020-02-14-21-42-04-133/source/sourcedir.tar.
gz
sagemaker-scikit-learn-2020-02-14-22-05-35-719/source/sourcedir.tar.
gz
sagemaker-scikit-learn-2020-02-14-22-19-55-787/source/sourcedir.tar.
gz
sagemaker-scikit-learn-2020-02-14-22-37-48-740/source/sourcedir.tar.
gz
sagemaker-scikit-learn-2020-02-14-22-44-28-251/source/sourcedir.tar.
gz
sagemaker-scikit-learn-2020-02-14-22-44-44-929/source/sourcedir.tar.
gz
sagemaker-scikit-learn-2020-02-14-22-52-19-766/source/sourcedir.tar.
gz
sagemaker-scikit-learn-2020-02-14-23-17-29-681/source/sourcedir.tar.
gz
sagemaker-scikit-learn-2020-02-14-23-54-36-424/source/sourcedir.tar.
gz
sagemaker-scikit-learn-2020-02-15-00-18-21-215/source/sourcedir.tar.
gz
Test passed!
```

Modeling

Now that you've uploaded your training data, it's time to define and train a model!

The type of model you create is up to you. For a binary classification task, you can choose to go one of three routes:

- Use a built-in classification algorithm, like LinearLearner.
- Define a custom Scikit-learn classifier, a comparison of models can be found here (https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html).
- Define a custom PyTorch neural network classifier.

It will be up to you to test out a variety of models and choose the best one. Your project will be graded on the accuracy of your final model.

EXERCISE: Complete a training script

To implement a custom classifier, you'll need to complete a `train.py` script. You've been given the folders `source_sklearn` and `source_pytorch` which hold starting code for a custom Scikit-learn model and a PyTorch model, respectively. Each directory has a `train.py` training script. To complete this project **you only need to complete one of these scripts**; the script that is responsible for training your final model.

A typical training script:

- Loads training data from a specified directory
- Parses any training & model hyperparameters (ex. nodes in a neural network, training epochs, etc.)
- Instantiates a model of your design, with any specified hyperparams
- Trains that model
- Finally, saves the model so that it can be hosted/deployed, later

Defining and training a model

Much of the training script code is provided for you. Almost all of your work will be done in the `if __name__ == '__main__':` section. To complete a `train.py` file, you will:

1. Import any extra libraries you need
2. Define any additional model training hyperparameters using `parser.add_argument`
3. Define a model in the `if __name__ == '__main__':` section
4. Train the model in that same section

Below, you can use `!pygmentize` to display an existing `train.py` file. Read through the code; all of your tasks are marked with `TODO` comments.

Note: If you choose to create a custom PyTorch model, you will be responsible for defining the model in the `model.py` file, and a `predict.py` file is provided. If you choose to use Scikit-learn, you only need a `train.py` file; you may import a classifier from the `sklearn` library.

In [5]:

```
# directory can be changed to: source_sklearn or source_pytorch
!pygmentize source_sklearn/train.py
```

```
from __future__ import print_function

import argparse
import os
import pandas as pd

from sklearn.externals import joblib

## TODO: Import any additional libraries you need to define a model

## We are going to use a classifier based on Gaussian Processes
from sklearn.svm import SVC
import sklearn.gaussian_process.kernels as gpkernels
from sklearn import metrics

# Provided model load function
def model_fn(model_dir):
    """Load model from the model_dir. This is the same model that is
saved
    in the main if statement.
    """
    print("Loading model.")

    # load using joblib
    model = joblib.load(os.path.join(model_dir, "model.joblib"))
    print("Done loading model.")

    return model

## TODO: Complete the main code
if __name__ == '__main__':

    # All of the model parameters and training parameters are sent a
s arguments
    # when this script is executed, during a training job

    # Here we set up an argument parser to easily access the paramet
ers
    parser = argparse.ArgumentParser()

    # SageMaker parameters, like the directories for training data a
nd saving models; set automatically
    # Do not need to change
    parser.add_argument('--output-data-dir', type=str, default=os.en
viron['SM_OUTPUT_DATA_DIR'])
    parser.add_argument('--model-dir', type=str, default=os.environ[
'SM_MODEL_DIR'])
    parser.add_argument('--data-dir', type=str, default=os.environ['
SM_CHANNEL_TRAIN'])
    parser.add_argument('--cparameter', type=float, default=1.0)
    parser.add_argument('--gamma', type=float, default=2.0)



    ## TODO: Add any additional arguments that you will need to pass
into your model

    # args holds all passed-in arguments
    args = parser.parse_args()

    # Read in csv training file
```

```
    training_dir = args.data_dir
    train_data = pd.read_csv(os.path.join(training_dir, "train.csv"
), header=None, names=None)

    # Labels are in the first column
    train_y = train_data.iloc[:,0]
    train_x = train_data.iloc[:,1:]


    ## --- Your code here --- ##

    c = args.cparameter
    gamma = args.gamma


    ## TODO: Define a model
    #print('We instiantate the model')
    model = SVC(C=c, gamma = gamma)


    model.fit(train_x, train_y)

    print('score-training {}'.format(model.score(train_x,train_y)))

    ## --- End of your code  --- ##


    # Save the trained model
    joblib.dump(model, os.path.join(args.model_dir, "model.joblib"))
```

Provided code

If you read the code above, you can see that the starter code includes a few things:

- Model loading (`model_fn`) and saving code
- Getting SageMaker's default hyperparameters
- Loading the training data by name, `train.csv` and extracting the features and labels, `train_x` , and `train_y`

If you'd like to read more about model saving with joblib for sklearn (https://scikit-learn.org/stable/modules/model_persistence.html) or with torch.save (https://pytorch.org/tutorials/beginner/saving_loading_models.html), click on the provided links.

Create an Estimator

When a custom model is constructed in SageMaker, an entry point must be specified. This is the Python file which will be executed when the model is trained; the `train.py` function you specified above. To run a custom training script in SageMaker, construct an estimator, and fill in the appropriate constructor arguments:

- **entry_point**: The path to the Python script SageMaker runs for training and prediction.
- **source_dir**: The path to the training script directory `source_sklearn` OR `source_pytorch`.
- **entry_point**: The path to the Python script SageMaker runs for training and prediction.
- **source_dir**: The path to the training script directory `train_sklearn` OR `train_pytorch`.
- **entry_point**: The path to the Python script SageMaker runs for training.
- **source_dir**: The path to the training script directory `train_sklearn` OR `train_pytorch`.
- **role**: Role ARN, which was specified, above.
- **train_instance_count**: The number of training instances (should be left at 1).
- **train_instance_type**: The type of SageMaker instance for training. Note: Because Scikit-learn does not natively support GPU training, Sagemaker Scikit-learn does not currently support training on GPU instance types.
- **sagemaker_session**: The session used to train on Sagemaker.
- **hyperparameters** (optional): A dictionary `{'name':value, ..}` passed to the train function as hyperparameters.

Note: For a PyTorch model, there is another optional argument **framework_version**, which you can set to the latest version of PyTorch, `1.0`.

EXERCISE: Define a Scikit-learn or PyTorch estimator

To import your desired estimator, use one of the following lines:

```
from sagemaker.sklearn.estimator import SKLearn
```

```
from sagemaker.pytorch import PyTorch
```

In [6]:

```
# your import and estimator code, here

from sagemaker.sklearn.estimator import SKLearn

output_path = 's3://{}/{}'.format(bucket, prefix)


SVCestimator = SKLearn(entry_point='train.py',
                       source_dir='source_sklearn', # this should be just "source"
 for your code
                       role=role,
                       train_instance_count=1,
                       train_instance_type='ml.c4.xlarge',
                       output_path=output_path,
                       sagemaker_session=sagemaker_session,
                       hyperparameters={
                           'cparameter':8.0,
                           'gamma':8.0
                       })
```

EXERCISE: Train the estimator

Train your estimator on the training data stored in S3. This should create a training job that you can monitor in your SageMaker console.

In [7]:

```
%%time

# Train your estimator on S3 training data

SVCestimator.fit({'train': input_data})
```

```
2020-02-15 09:34:28 Starting - Starting the training job...
2020-02-15 09:34:29 Starting - Launching requested ML instance
s......
2020-02-15 09:35:32 Starting - Preparing the instances for trainin
g......
2020-02-15 09:36:33 Downloading - Downloading input data...
2020-02-15 09:37:27 Training - Training image download completed. Tr
aining in progress.
2020-02-15 09:37:27 Uploading - Uploading generated training model
2020-02-15 09:37:27 Completed - Training job completed
2020-02-15 09:37:15,963 sagemaker-containers INFO     Imported frame
work sagemaker_sklearn_container.training
2020-02-15 09:37:15,965 sagemaker-containers INFO     No GPUs detect
ed (normal if no gpus installed)
2020-02-15 09:37:15,975 sagemaker_sklearn_container.training INFO
Invoking user training script.
2020-02-15 09:37:16,429 sagemaker-containers INFO     Module train d
oes not provide a setup.py.
Generating setup.py
2020-02-15 09:37:16,429 sagemaker-containers INFO     Generating set
up.cfg
2020-02-15 09:37:16,429 sagemaker-containers INFO     Generating MAN
IFEST.in
2020-02-15 09:37:16,430 sagemaker-containers INFO     Installing mod
ule with the following command:
/miniconda3/bin/python -m pip install .
Processing /opt/ml/code
Building wheels for collected packages: train
  Building wheel for train (setup.py): started
  Building wheel for train (setup.py): finished with status 'done'
  Created wheel for train: filename=train-1.0.0-py2.py3-none-any.whl
size=9328 sha256=50303c0e70dd2705dc795d369e8f76d65e4746aeef5a574ea41
448de04b701fd
  Stored in directory: /tmp/pip-ephem-wheel-cache-e8ak2b15/wheels/3
5/24/16/37574d11bf9bde50616c67372a334f94fa8356bc7164af8ca3
Successfully built train
Installing collected packages: train
Successfully installed train-1.0.0
2020-02-15 09:37:17,843 sagemaker-containers INFO     No GPUs detect
ed (normal if no gpus installed)
2020-02-15 09:37:17,853 sagemaker-containers INFO     Invoking user
 script

Training Env:

{
    "additional_framework_parameters": {},
    "channel_input_dirs": {
        "train": "/opt/ml/input/data/train"
    },
    "current_host": "algo-1",
    "framework_module": "sagemaker_sklearn_container.training:main",
    "hosts": [
        "algo-1"
    ],
    "hyperparameters": {
        "cparameter": 8.0,
        "gamma": 8.0
    },
    "input_config_dir": "/opt/ml/input/config",
    "input_data_config": {
```

```
            "train": {
                "TrainingInputMode": "File",
                "S3DistributionType": "FullyReplicated",
                "RecordWrapperType": "None"
            }
        },
        "input_dir": "/opt/ml/input",
        "is_master": true,
        "job_name": "sagemaker-scikit-learn-2020-02-15-09-34-28-097",
        "log_level": 20,
        "master_hostname": "algo-1",
        "model_dir": "/opt/ml/model",
        "module_dir": "s3://sagemaker-us-east-1-415235263340/sagemaker-s
    cikit-learn-2020-02-15-09-34-28-097/source/sourcedir.tar.gz",
        "module_name": "train",
        "network_interface_name": "eth0",
        "num_cpus": 4,
        "num_gpus": 0,
        "output_data_dir": "/opt/ml/output/data",
        "output_dir": "/opt/ml/output",
        "output_intermediate_dir": "/opt/ml/output/intermediate",
        "resource_config": {
            "current_host": "algo-1",
            "hosts": [
                "algo-1"
            ],
            "network_interface_name": "eth0"
        },
        "user_entry_point": "train.py"
    }

    Environment variables:

    SM_HOSTS=["algo-1"]
    SM_NETWORK_INTERFACE_NAME=eth0
    SM_HPS={"cparameter":8.0,"gamma":8.0}
    SM_USER_ENTRY_POINT=train.py
    SM_FRAMEWORK_PARAMS={}
    SM_RESOURCE_CONFIG={"current_host":"algo-1","hosts":["algo-1"],"netw
    ork_interface_name":"eth0"}
    SM_INPUT_DATA_CONFIG={"train":{"RecordWrapperType":"None","S3Distrib
    utionType":"FullyReplicated","TrainingInputMode":"File"}}
    SM_OUTPUT_DATA_DIR=/opt/ml/output/data
    SM_CHANNELS=["train"]
    SM_CURRENT_HOST=algo-1
    SM_MODULE_NAME=train
    SM_LOG_LEVEL=20
    SM_FRAMEWORK_MODULE=sagemaker_sklearn_container.training:main
    SM_INPUT_DIR=/opt/ml/input
    SM_INPUT_CONFIG_DIR=/opt/ml/input/config
    SM_OUTPUT_DIR=/opt/ml/output
    SM_NUM_CPUS=4
    SM_NUM_GPUS=0
    SM_MODEL_DIR=/opt/ml/model
    SM_MODULE_DIR=s3://sagemaker-us-east-1-415235263340/sagemaker-scikit
    -learn-2020-02-15-09-34-28-097/source/sourcedir.tar.gz
    SM_TRAINING_ENV={"additional_framework_parameters":{},"channel_input
    _dirs":{"train":"/opt/ml/input/data/train"},"current_host":"algo-
    1","framework_module":"sagemaker_sklearn_container.training:main","h
    osts":["algo-1"],"hyperparameters":{"cparameter":8.0,"gamma":8.0},"i
    nput_config_dir":"/opt/ml/input/config","input_data_config":{"trai
```

```
n":{"RecordWrapperType":"None","S3DistributionType":"FullyReplicate
d","TrainingInputMode":"File"}},"input_dir":"/opt/ml/input","is_mast
er":true,"job_name":"sagemaker-scikit-learn-2020-02-15-09-34-28-09
7","log_level":20,"master_hostname":"algo-1","model_dir":"/opt/ml/mo
del","module_dir":"s3://sagemaker-us-east-1-415235263340/sagemaker-s
cikit-learn-2020-02-15-09-34-28-097/source/sourcedir.tar.gz","module
_name":"train","network_interface_name":"eth0","num_cpus":4,"num_gpu
s":0,"output_data_dir":"/opt/ml/output/data","output_dir":"/opt/ml/o
utput","output_intermediate_dir":"/opt/ml/output/intermediate","reso
urce_config":{"current_host":"algo-1","hosts":["algo-1"],"network_in
terface_name":"eth0"},"user_entry_point":"train.py"}
SM_USER_ARGS=["--cparameter","8.0","--gamma","8.0"]
SM_OUTPUT_INTERMEDIATE_DIR=/opt/ml/output/intermediate
SM_CHANNEL_TRAIN=/opt/ml/input/data/train
SM_HP_CPARAMETER=8.0
SM_HP_GAMMA=8.0
PYTHONPATH=/miniconda3/bin:/miniconda3/lib/python37.zip:/miniconda3/
lib/python3.7:/miniconda3/lib/python3.7/lib-dynload:/miniconda3/lib/
python3.7/site-packages

Invoking script with the following command:

/miniconda3/bin/python -m train --cparameter 8.0 --gamma 8.0


/miniconda3/lib/python3.7/site-packages/sklearn/externals/joblib/ext
ernals/cloudpickle/cloudpickle.py:47: DeprecationWarning: the imp mo
dule is deprecated in favour of importlib; see the module's document
ation for alternative uses
  import imp
score-training 0.9857142857142858
2020-02-15 09:37:19,738 sagemaker-containers INFO     Reporting trai
ning SUCCESS
Training seconds: 54
Billable seconds: 54
CPU times: user 433 ms, sys: 24.1 ms, total: 457 ms
Wall time: 3min 11s
```

Hyperparameter tuning

Let us see if with hyperparameter tuning we can get somewhere!

In [7]:

```
from sklearn import metrics

SVCestimator = SKLearn(entry_point='train.py',
                       source_dir='source_sklearn', # this should be just "source"
 for your code
                       role=role,
                       train_instance_count=1,
                       train_instance_type='ml.c4.xlarge',
                       output_path=output_path,
                       sagemaker_session=sagemaker_session,
                       hyperparameters={
                           'cparameter':1.0,
                           'gamma':2.0
                       })
```

In [8]:

```
from sagemaker.tuner import HyperparameterTuner, IntegerParameter, ContinuousPar
ameter, CategoricalParameter

hyperparameter_ranges = { 'cparameter': ContinuousParameter(0.1, 10.0),
                          'gamma': ContinuousParameter(0.5, 10.0)}


SVCestimator_tuner = HyperparameterTuner(estimator = SVCestimator,
                                         objective_metric_name = 'score-training',
                                         objective_type = 'Maximize',
                                         max_jobs = 10,
                                         max_parallel_jobs = 3,
                                         hyperparameter_ranges = hyperparameter_ra
nges,
                                         metric_definitions = [{'Name':'score-trai
ning',
                                             'Regex': 'score-training ([0-9\\.]+)'
}]
                                         )
```

In [11]:

```
SVCestimator_tuner.fit({'train': input_data})
```

In [12]:

```
SVCestimator_tuner.wait()
```

```
.....................................................................
```

In [13]:

```
SVCestimator_tuner.best_training_job()
```

Out[13]:

```
'sagemaker-scikit-lea-200215-0018-002-7d418a49'
```

In [14]:

```
SVC_attached = sagemaker.estimator.Estimator.attach(SVCestimator_tuner.best_training_job())
```

```
2020-02-15 00:26:06 Starting - Preparing the instances for training
2020-02-15 00:26:06 Downloading - Downloading input data
2020-02-15 00:26:06 Training - Training image download completed. Tr
aining in progress.
2020-02-15 00:26:06 Uploading - Uploading generated training model
2020-02-15 00:26:06 Completed - Training job completed2020-02-15 00:
25:12,981 sagemaker-containers INFO     Imported framework sagemaker
_sklearn_container.training
2020-02-15 00:25:12,982 sagemaker-containers INFO     Failed to pars
e hyperparameter _tuning_objective_metric value score-training to Js
on.
Returning the value itself
2020-02-15 00:25:12,984 sagemaker-containers INFO     No GPUs detect
ed (normal if no gpus installed)
2020-02-15 00:25:12,994 sagemaker_sklearn_container.training INFO
Invoking user training script.
2020-02-15 00:25:13,358 sagemaker-containers INFO     Module train d
oes not provide a setup.py.
Generating setup.py
2020-02-15 00:25:13,359 sagemaker-containers INFO     Generating set
up.cfg
2020-02-15 00:25:13,359 sagemaker-containers INFO     Generating MAN
IFEST.in
2020-02-15 00:25:13,359 sagemaker-containers INFO     Installing mod
ule with the following command:
/miniconda3/bin/python -m pip install .
Processing /opt/ml/code
Building wheels for collected packages: train
  Building wheel for train (setup.py): started
  Building wheel for train (setup.py): finished with status 'done'
  Created wheel for train: filename=train-1.0.0-py2.py3-none-any.whl
size=9329 sha256=8eb4bf7164d8992fd90a3a8dd57be9ba00729b2243d08029d99
c06764d9ea654
  Stored in directory: /tmp/pip-ephem-wheel-cache-mv8a8_j9/wheels/3
5/24/16/37574d11bf9bde50616c67372a334f94fa8356bc7164af8ca3
Successfully built train
Installing collected packages: train
Successfully installed train-1.0.0
2020-02-15 00:25:14,850 sagemaker-containers INFO     Failed to pars
e hyperparameter _tuning_objective_metric value score-training to Js
on.
Returning the value itself
2020-02-15 00:25:14,853 sagemaker-containers INFO     No GPUs detect
ed (normal if no gpus installed)
2020-02-15 00:25:14,864 sagemaker-containers INFO     Invoking user
 script

Training Env:

{
    "additional_framework_parameters": {
        "sagemaker_estimator_class_name": "SKLearn",
        "sagemaker_estimator_module": "sagemaker.sklearn.estimator"
    },
    "channel_input_dirs": {
        "train": "/opt/ml/input/data/train"
    },
    "current_host": "algo-1",
    "framework_module": "sagemaker_sklearn_container.training:main",
    "hosts": [
        "algo-1"
```

```
    ],
    "hyperparameters": {
        "cparameter": 9.094539644798147,
        "gamma": 3.3955752760882283
    },
    "input_config_dir": "/opt/ml/input/config",
    "input_data_config": {
        "train": {
            "TrainingInputMode": "File",
            "S3DistributionType": "FullyReplicated",
            "RecordWrapperType": "None"
        }
    },
    "input_dir": "/opt/ml/input",
    "is_master": true,
    "job_name": "sagemaker-scikit-lea-200215-0018-002-7d418a49",
    "log_level": 20,
    "master_hostname": "algo-1",
    "model_dir": "/opt/ml/model",
    "module_dir": "s3://sagemaker-us-east-1-415235263340/sagemaker-s
cikit-learn-2020-02-15-00-18-21-215/source/sourcedir.tar.gz",
    "module_name": "train",
    "network_interface_name": "eth0",
    "num_cpus": 4,
    "num_gpus": 0,
    "output_data_dir": "/opt/ml/output/data",
    "output_dir": "/opt/ml/output",
    "output_intermediate_dir": "/opt/ml/output/intermediate",
    "resource_config": {
        "current_host": "algo-1",
        "hosts": [
            "algo-1"
        ],
        "network_interface_name": "eth0"
    },
    "user_entry_point": "train.py"
}

Environment variables:

SM_HOSTS=["algo-1"]
SM_NETWORK_INTERFACE_NAME=eth0
SM_HPS={"cparameter":9.094539644798147,"gamma":3.3955752760882283}
SM_USER_ENTRY_POINT=train.py
SM_FRAMEWORK_PARAMS={"sagemaker_estimator_class_name":"SKLearn","sag
emaker_estimator_module":"sagemaker.sklearn.estimator"}
SM_RESOURCE_CONFIG={"current_host":"algo-1","hosts":["algo-1"],"netw
ork_interface_name":"eth0"}
SM_INPUT_DATA_CONFIG={"train":{"RecordWrapperType":"None","S3Distrib
utionType":"FullyReplicated","TrainingInputMode":"File"}}
SM_OUTPUT_DATA_DIR=/opt/ml/output/data
SM_CHANNELS=["train"]
SM_CURRENT_HOST=algo-1
SM_MODULE_NAME=train
SM_LOG_LEVEL=20
SM_FRAMEWORK_MODULE=sagemaker_sklearn_container.training:main
SM_INPUT_DIR=/opt/ml/input
SM_INPUT_CONFIG_DIR=/opt/ml/input/config
SM_OUTPUT_DIR=/opt/ml/output
SM_NUM_CPUS=4
SM_NUM_GPUS=0
```

```
SM_MODEL_DIR=/opt/ml/model
SM_MODULE_DIR=s3://sagemaker-us-east-1-415235263340/sagemaker-scikit
-learn-2020-02-15-00-18-21-215/source/sourcedir.tar.gz
SM_TRAINING_ENV={"additional_framework_parameters":{"sagemaker_estim
ator_class_name":"SKLearn","sagemaker_estimator_module":"sagemaker.s
klearn.estimator"},"channel_input_dirs":{"train":"/opt/ml/input/dat
a/train"},"current_host":"algo-1","framework_module":"sagemaker_skle
arn_container.training:main","hosts":["algo-1"],"hyperparameters":
{"cparameter":9.094539644798147,"gamma":3.3955752760882283},"input_c
onfig_dir":"/opt/ml/input/config","input_data_config":{"train":{"Rec
ordWrapperType":"None","S3DistributionType":"FullyReplicated","Train
ingInputMode":"File"}},"input_dir":"/opt/ml/input","is_master":tru
e,"job_name":"sagemaker-scikit-lea-200215-0018-002-7d418a49","log_le
vel":20,"master_hostname":"algo-1","model_dir":"/opt/ml/model","modu
le_dir":"s3://sagemaker-us-east-1-415235263340/sagemaker-scikit-lear
n-2020-02-15-00-18-21-215/source/sourcedir.tar.gz","module_name":"tr
ain","network_interface_name":"eth0","num_cpus":4,"num_gpus":0,"outp
ut_data_dir":"/opt/ml/output/data","output_dir":"/opt/ml/output","ou
tput_intermediate_dir":"/opt/ml/output/intermediate","resource_confi
g":{"current_host":"algo-1","hosts":["algo-1"],"network_interface_na
me":"eth0"},"user_entry_point":"train.py"}
SM_USER_ARGS=["--cparameter","9.094539644798147","--gamma","3.395575
2760882283"]
SM_OUTPUT_INTERMEDIATE_DIR=/opt/ml/output/intermediate
SM_CHANNEL_TRAIN=/opt/ml/input/data/train
SM_HP_CPARAMETER=9.094539644798147
SM_HP_GAMMA=3.3955752760882283
PYTHONPATH=/miniconda3/bin:/miniconda3/lib/python37.zip:/miniconda3/
lib/python3.7:/miniconda3/lib/python3.7/lib-dynload:/miniconda3/lib/
python3.7/site-packages

Invoking script with the following command:

/miniconda3/bin/python -m train --cparameter 9.094539644798147 --gam
ma 3.3955752760882283


/miniconda3/lib/python3.7/site-packages/sklearn/externals/joblib/ext
ernals/cloudpickle/cloudpickle.py:47: DeprecationWarning: the imp mo
dule is deprecated in favour of importlib; see the module's document
ation for alternative uses
  import imp
score-training 0.9857142857142858
2020-02-15 00:25:16,807 sagemaker-containers INFO     Reporting trai
ning SUCCESS
Training seconds: 343
Billable seconds: 343
```

EXERCISE: Deploy the trained model

After training, deploy your model to create a `predictor`. If you're using a PyTorch model, you'll need to create a trained `PyTorchModel` that accepts the trained `<model>.model_data` as an input parameter and points to the provided `source_pytorch/predict.py` file as an entry point.

To deploy a trained model, you'll use `<model>.deploy`, which takes in two arguments:

- **initial_instance_count**: The number of deployed instances (1).
- **instance_type**: The type of SageMaker instance for deployment.

Note: If you run into an instance error, it may be because you chose the wrong training or deployment instance_type. It may help to refer to your previous exercise code to see which types of instances we used.

In [8]:

```
%%time

# uncomment, if needed
# from sagemaker.pytorch import PyTorchModel

# deploy your model to create a predictor

# predictor = SVC_attached.deploy(instance_type="ml.c4.xlarge", initial_instance
_count=1
#             )

predictor = SVCestimator.deploy(instance_type="ml.c4.xlarge", initial_instance_c
ount=1
            )
```

```
-----------------!CPU times: user 302 ms, sys: 22.6 ms, total: 324 m
s
Wall time: 8min 32s
```

Evaluating Your Model

Once your model is deployed, you can see how it performs when applied to our test data.

The provided cell below, reads in the test data, assuming it is stored locally in `data_dir` and named `test.csv`. The labels and features are extracted from the `.csv` file.

In [9]:

```
"""
DON'T MODIFY ANYTHING IN THIS CELL THAT IS BELOW THIS LINE
"""
import os

# read in test data, assuming it is stored locally
test_data = pd.read_csv(os.path.join(data_dir, "test.csv"), header=None, names=N
one)

# labels are in the first column
test_y = test_data.iloc[:,0]
test_x = test_data.iloc[:,1:]
```

EXERCISE: Determine the accuracy of your model

Use your deployed `predictor` to generate predicted, class labels for the test data. Compare those to the *true* labels, `test_y` , and calculate the accuracy as a value between 0 and 1.0 that indicates the fraction of test data that your model classified correctly. You may use sklearn.metrics (https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics) for this calculation.

To pass this project, your model should get at least 90% test accuracy.

In [10]:

```
# First: generate predicted, class labels
test_y_preds = predictor.predict(test_x)


"""
DON'T MODIFY ANYTHING IN THIS CELL THAT IS BELOW THIS LINE
"""
# test that your model generates the correct number of labels
assert len(test_y_preds)==len(test_y), 'Unexpected number of predictions.'
print('Test passed!')
```

```
Test passed!
```

In [11]:

```
# Second: calculate the test accuracy
from sklearn import metrics
accuracy = metrics.accuracy_score(test_y, test_y_preds)

print(accuracy)


## print out the array of predicted and true labels, if you want
print('\nPredicted class labels: ')
print(test_y_preds)
print('\nTrue class labels: ')
print(test_y.values)
```

```
0.92

Predicted class labels:
[0 0 0 0 0 0 1 1 1 1 1 1 0 0 1 0 1 1 1 0 0 1 0 1 1]

True class labels:
[0 0 0 0 0 0 1 1 1 1 1 1 0 0 1 0 1 1 1 1 0 1 1 1 1]
```

In [12]:

```
train_data = pd.read_csv(os.path.join(data_dir, "train.csv"), header=None, names
=None)

# labels are in the first column
train_y = train_data.iloc[:,0]
train_x = train_data.iloc[:,1:]
```

In [13]:

```
train_y_preds = predictor.predict(train_x)
accuracy = metrics.accuracy_score(train_y, train_y_preds)
print(accuracy)

print('\nPredicted class labels: ')
print(train_y_preds)
print('\nTrue class labels: ')
print(train_y.values)
```

```
0.9857142857142858

Predicted class labels:
[0 1 1 1 0 0 1 1 1 1 0 1 1 1 1 1 0 1 1 1 0 1 1 1 0 0 0 1 1 1 1 1 0 0
1 0 0
 0 1 0 1 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 1 0 0 1 1 0 0 0]

True class labels:
[0 1 1 1 0 0 1 1 1 1 0 1 1 1 1 1 0 1 1 1 0 1 1 1 0 0 0 1 1 1 1 1 0 0
1 0 0
 0 1 0 1 1 1 1 1 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 1 0 0 1 1 0 0 0]
```

In [ ]:

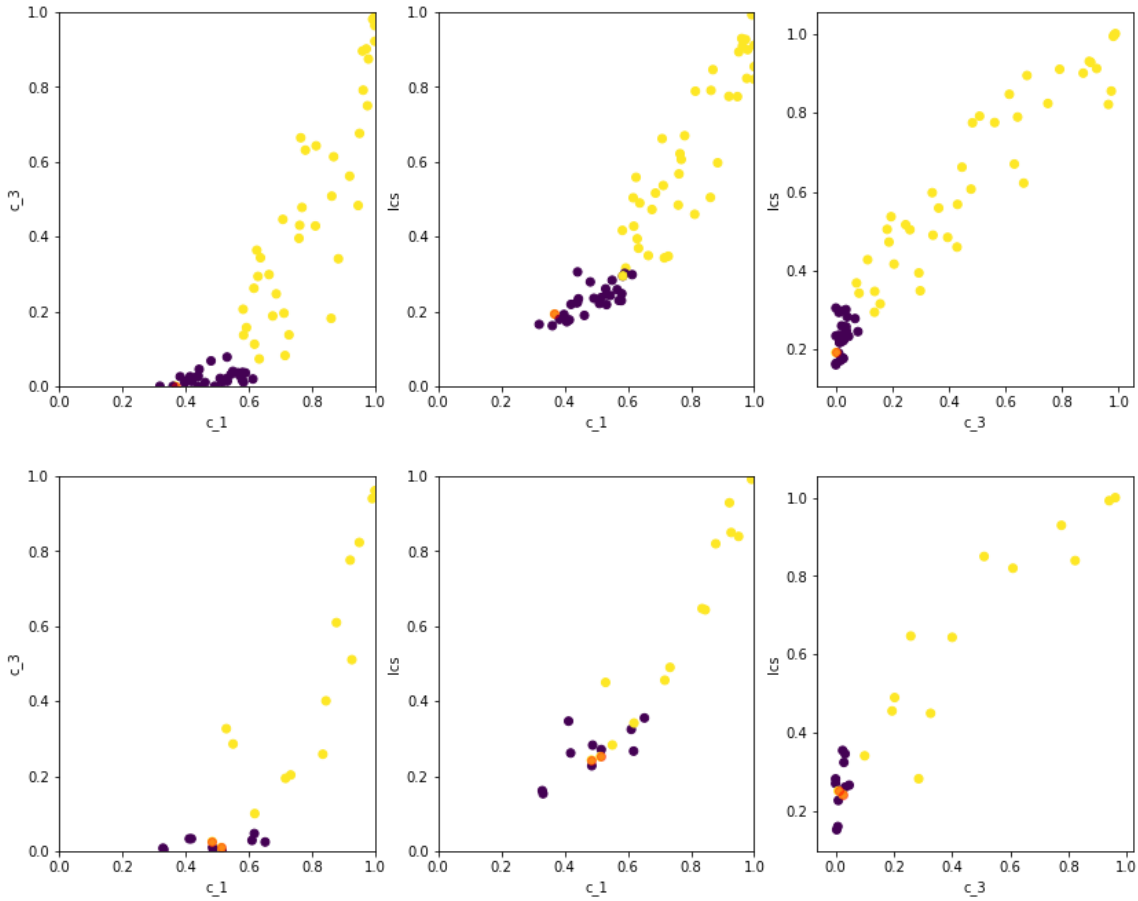
In [14]:

```
import matplotlib.pylab as plt
%matplotlib inline
```

In [20]:

```
for tempx,tempy,tempy_pred in [(train_x,train_y,train_y_preds), (test_x,test_y,
test_y_preds)]:
    fig, ax = plt.subplots(1,3,figsize = (14,5))
    print(tempx.shape)
    ax[0].scatter(tempx.values[:,0],tempx.values[:,1],c = tempy)
    ax[1].scatter(tempx.values[:,0],tempx.values[:,2],c = tempy)
    ax[2].scatter(tempx.values[:,1],tempx.values[:,2],c = tempy)

    sel = tempy_pred!=tempy
    ax[0].scatter(tempx.values[sel,0],tempx.values[sel,1],c = 'r', alpha = 0.4)
    ax[1].scatter(tempx.values[sel,0],tempx.values[sel,2],c = 'r', alpha = 0.4)
    ax[2].scatter(tempx.values[sel,1],tempx.values[sel,2],c = 'r', alpha = 0.4)

    ax[0].set_xlabel('c_1')
    ax[0].set_ylabel('c_3')

    ax[1].set_xlabel('c_1')
    ax[1].set_ylabel('lcs')

    ax[2].set_xlabel('c_3')
    ax[2].set_ylabel('lcs')
    ax[0].set_title('')
    for i in range(2):
        ax[i].set_xlim(0,1)
        ax[i].set_ylim(0,1)
```

```
(70, 3)
(25, 3)
```



Question 1: How many false positives and false negatives did your model produce, if any? And why do you think this is?

Answer: The RBF SVM has not produced any false positives in this dataset. However, 8% of the negative predictions were wrong (orange points in figure above). A first thought would be to think that there is some sort of overfitting, since I searched in the hyperparameter space using the training set score as an objective. However, by looking at the position of these misclassified points in the three possible projected planes, it seems clear that overfitting is not the only answer. By a careful observation of other c_n (in the other notebook), it seems that just by substituting c_3 or c_1 it would not be enough to separate them and we need to provide another sort of information. In other words, we would need to add another quantity to distinguish them properly: c_1, c_3 and lcs are not enough.

In [17]:

```
falsepositives = ((test_y==0)*(test_y_preds>0)).sum()/test_y.shape[0]
falsenegatives = ((test_y>0)*(test_y_preds==0)).sum()/test_y.shape[0]
print('The fraction of false positives is ', falsepositives)
print('The fraction of false negative is ', falsenegatives)
```

```
The fraction of false positives is  0.0
The fraction of false negative is  0.08
```

```
/home/ec2-user/anaconda3/envs/pytorch_p36/lib/python3.6/site-package
s/pandas/core/computation/expressions.py:183: UserWarning: evaluatin
g in Python space because the '*' operator is not supported by numex
pr for the bool dtype, use '&' instead
  .format(op=op_str, alt_op=unsupported[op_str]))
```

Question 2: How did you decide on the type of model to use?

Answer: The boundary between the two datasets seemed relatively non-linear, and by checking the performance of different algorithms (https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html), I thought that the best for this case would be either a Gaussian Process classifier, k-means algorithm or this type of Support Vector Machine. I tried first with GP but I could not fit properly the training data set. The reason behind this was actually that there was a bug in the previous notebook, but I was not aware of it and decided to change algorithm to SVM. It was then when I realized the problem was not in the algorithm itself but in the data preparation. However I had already wrote all the code for SVM and its performance is pretty good.

EXERCISE: Clean up Resources

After you're done evaluating your model, **delete your model endpoint**. You can do this with a call to `.delete_endpoint()`. You need to show, in this notebook, that the endpoint was deleted. Any other resources, you may delete from the AWS console, and you will find more instructions on cleaning up all your resources, below.

In [21]:

```
# uncomment and fill in the line below!
# <name_of_deployed_predictor>.delete_endpoint()
predictor.delete_endpoint()
```

Deleting S3 bucket

When you are *completely* done with training and testing models, you can also delete your entire S3 bucket. If you do this before you are done training your model, you'll have to recreate your S3 bucket and upload your training data again.

In [ ]:

```
# deleting bucket, uncomment lines below

bucket_to_delete = boto3.resource('s3').Bucket(bucket)
bucket_to_delete.objects.all().delete()
```

Deleting all your models and instances

When you are *completely* done with this project and do **not** ever want to revisit this notebook, you can choose to delete all of your SageMaker notebook instances and models by following [these instructions (https://docs.aws.amazon.com/sagemaker/latest/dg/ex1-cleanup.html)](https://docs.aws.amazon.com/sagemaker/latest/dg/ex1-cleanup.html). Before you delete this notebook instance, I recommend at least downloading a copy and saving it, locally.

Further Directions

There are many ways to improve or add on to this project to expand your learning or make this more of a unique project for you. A few ideas are listed below:

- Train a classifier to predict the *category* (1-3) of plagiarism and not just plagiarized (1) or not (0).
- Utilize a different and larger dataset to see if this model can be extended to other types of plagiarism.
- Use language or character-level analysis to find different (and more) similarity features.
- Write a complete pipeline function that accepts a source text and submitted text file, and classifies the submitted text as plagiarized or not.
- Use API Gateway and a lambda function to deploy your model to a web application.

These are all just options for extending your work. If you've completed all the exercises in this notebook, you've completed a real-world application, and can proceed to submit your project. Great job!