DATA SCIENCE DIVISION FELLOWSHIP PROGRAM: TAKEHOME EXAM

STATISTICS CANADA

ASSIGNMENT: MACHINE LEARNING

You have **until 6:00 PM on December 3rd, 2022, to complete and submit your exam.**

Return the completed exam to: statcan.fellowship-fellowship.statcan@statcan.gc.ca.

The date and time on your e-mail will serve as proof that you completed the exam within the allotted time. No delays will be accepted. Therefore, if the exam is not submitted by the due date; your candidature won't be considered further within this process.

All tasks, within an assignment, must be completed for your submission to be assessed and evaluated as part of this selection process.

The content of this exam is confidential. Do not divulge the content of the exam with others.

PURPOSE

The purpose of the take-home exam is to showcase your skills. We will assess your ability to apply common data science and machine learning related techniques in Python, convey results in a coherent and understandable manner, and create appropriate visualizations to present and interpret the data.

The exam will assess the following skills and competencies:

- Programming and machine learning skills
- Data management
- Analytical thinking
- Storytelling and data visualization
- Communication

Carefully read all instructions below before beginning.

INSTRUCTIONS

Two different assignments are provided. **Complete one (1)** of the two assignments.

Each assignment involves accessing a publicly available dataset and performing a sequence of tasks.

Some tasks are structured and specific while others are more open. Links are provided for the datasets. If the links do not work, it is your duty to find the dataset through some other means.

WHAT TO SUBMIT

When you have completed the assignment, you must provide all code for the assignment tasks. This includes all code used to complete the tasks and other auxiliary code required. Feel free to include code for any data exploration that you may choose to perform.

Code should be in the form of a Jupyter notebook (or equivalent) and should be well documented. We should not be required to run your code to produce figures, meaning the cells of your notebook should have already been run when viewed on Gitlab/Github. You will be required to create a public Git repository (repo) (Gitlab/Github) and place your Jupyter notebook in there.

Your submission will involve emailing us the link to your Git repository, which will allow us to evaluate the code, figures and documentation within the repo.

We are not interested in you producing a model with 99% validation accuracy, so please, do not spend time and effort on **any** hyper-parameter tuning. Rather, we are interested in your thought process, your ability to explain what you did, why you did it, and given the constraints of the assignment, how you may have done things differently. Be prepared to answer questions related to this assignment during the interview.

Here are some other things we are looking for in your submission:

- Reproducibility
- Maintainability of the solution
- Following PEP8 / Google / Numpy style guide
- Choosing meaningful names for functions and variables
- Good coding practices

IMPORTANT NOTES

- You must complete the exam independently and without assistance.
- Ensure the Git repo you set up and submit cannot be edited by another individual and can be access by the selection committee (public).
- Submissions assessed will be picked up in the Git repo no later than 6:00 pm (EST) on Saturday, December 2, 2022.

Good luck and have fun!

ASSIGNMENT 1: CARS DATASET

Dataset

- The Cars Dataset can be found here: (https://ai.stanford.edu/~jkrause/cars/car_dataset.html)
- The *Cars* dataset contains 16,185 images of 196 classes of cars. The data is split into 8,144 training images and 8,041 testing images, where each class has been split roughly in a 50-50 split.

Objective

- Your clients would like a model that can classify cars found from images taken from traffic cameras.
- This assignment will involve exploring the importance of the size of a labelled dataset for supervised learning.

Note: For questions where you are asked to explain or interpret results, please do so within the Jupyter notebook (or equivalent) using Markdown

TASK 1 - *Build a function that converts a labelled dataset into labelled and unlabelled subsets*.

☐	Build a function with the following inputs/arguments: dataset_labels (list of integers), proportion (float between [0,1]). Here, the indices of dataset_labels represent the indices of the instances in your dataset, and dataset_labels[idx] represents the class index of data instance idx.
☐	This function “removes” the labels from (100 * proportion)% of the original dataset. Once this proportion of instances from dataset have been sampled, these labels should be removed so that these instances should be treated as an unlabelled dataset.
☐	The function should ensure that all classes have at least 1 instance labelled within the dataset.
☐	What the function returns is up to you. However, from the output of this function you should know: • Which data instances (indices) in your dataset you know are labelled or not labelled • If a data instance is labelled, you should know its label • If a data instance is unlabelled, you should still know what the original label was.

TASK 2 - *Data cleaning*

☐	Iterate through the images within the dataset and remove from disk (delete) any instances that are not RGB. Specifically, delete any images from disk that do not have 3 channels.

TASK 3 - *Dataset representation*

☐	Create an empty dictionary that maps indices of instances in your dataset to a dictionary with this structure (dataset of size N instances): {1: {'embedding': <np.ndarray>, 'class_idx': <int>, ‘labelled': <boolean or int>}, 2: {...},

	…, N: {…}}
□	Using PyTorch, load a pre-trained resnet18, setting it to the variable: model. We will use model as a feature embedder. The final layer in the resnet18 (model.fc) is a nn.Linear(*, 1000), for some embedding dimension *. Replace model.fc with an Identity layer, so that the forward pass will return the output from the final pooling layer, rather than the logits. For each image in the dataset, load the image from disk and pass it through the pre-trained resnet18 to receive the embeddings. Then, update the dict with the newly acquired embedding and the class_idx. Init labelled to True (or 1). Beware of data types for your tensor!
□	Once done, save this dictionary to disk using torch.save(). This is your full datasest and will be used for all subsequent steps.

TASK 4 - *Build a partially labelled dataset*

□	Using the function from Task 1, and the processed dataset from Task 3, create a dataset with only 40% of the data being labelled, and 60% unlabelled.

TASK 5 - *Create train/validation split*

□	Create a function that takes two arguments, dataset_inputs and dataset_labels, and a float between [0,1], “training_proportion”. This function will take a dataset (inputs and labels) and split it into training and validation by the training_proportion. The function returns 4 objects, training_inputs, training_labels, test_inputs, test_labels.
□	Note: feel free to make use of any packages/libraries (e.g. scikit learn)

TASK 6 - Create experiment(s) to convince clients that more labelled data will improve model performance

□	It is your job to convince your clients to label some more data, since you believe that the model will perform better with more labelled data, however, your clients are busy and labelling additional data is time consuming and costly.
□	For all models trained, use a simple linear model using the embeddings as your inputs using sklearn.linear_model.SGDClassifier. Use whatever number of iterations it takes to train a model in 1 minute MAX (don't worry, we aren't judging results based on accuracy).
□	Create some sort of figure(s) to show your client, based on test(s) and/or experiment(s) you have performed that would convince them that labelling more data will lead to a better classifier for this use-case. Make sure to use your function created in Task 5. Analyze the plot(s) produced and explain them to your clients. Remember, in this scenario you only have 40% of the data that is labelled, with the remaining 60% of the data being unlabelled.

TASK 7 - *Active learning to select new instances to be labelled*

□	Your clients have agreed to label an additional 25% of the dataset (to a total of 65% of the original dataset).

☐	Using the model trained on 40% of the data and the unlabelled inputs, use predict_proba function of your sklearn.linear_model.SGDClassifier model to get the probability scores for each class on your unlabelled instances. Use from scipy.stats entropy to compute the entropy of the predictions for the unlabelled instances. Select the K instances with the highest entropy for labelling, where K is the remaining number of instances that will bring your total labelled dataset to 65%. This will be added to the labelled subset and will be your 'final' dataset.
☐	NOTE: Since the original dataset is fully labelled, just use the true labels from the original dataset. Hence, keeping track of indices of instances will have been important for the earlier instructions.

TASK 8 - *Final model training and evaluation*

☐	You now have additional labelled data.
☐	Using your training dataset, train a final model and then use it to evaluate your model on your hold out validation dataset.
☐	Explain your results and your implementation.
☐	Based on your evaluation, was the decision to label more instances a good one?
☐	Report the final performance.

TASK 9 – *Submit exam to Statistics Canada*

☐	See instructions under "What to Submit" on the first page of this document. Submit your exam at statcan.fellowship-fellowship.statcan@statcan.gc.ca Include: - all code used to complete the task and other required auxiliary code - share the link to your public Git repository so we can evaluate the code, figures and documentation

ASSIGNMENT 2: NATURAL DISASTERS DATASET

Dataset

- The Natural Disasters Dataset can be found here: https://www.kaggle.com/competitions/nlp-getting-started/
- This is a text (NLP) dataset that contains tweet about real disasters, or not (binary classification).

Objective

- Your clients would like to classify tweets as either being about a real disaster or not.
- They would like you to use the simplest possible model that "gets the job done" and achieves sufficiently high accuracy (a number they don't want to define). As well, they value understanding how the model arrived at its classification, if possible.
- You will train and evaluate 3 learning algorithms of increasing complexity.
- Each learning algorithm will require preprocessing the text data differently, which you will have to implement yourself.
- Your goal is to present to your clients options that balance explainable models with performance.

Note: For questions where you are asked to explain or interpret results, please do so within the Jupyter notebook (or equivalent) using Markdown.

TASK 1 - *Bag of words model*

□	You will build a bag of words model.
□	Perform all of the pre-processing of the dataset to prepare it for a bag of words model. Explain all of your choices around pre-processing.
□	Train the model and provide plots on the evaluation.

TASK 2 - *Feature generation and traditional ML model*

□	You will now use TF-IDF to generate features for your dataset. Moreover, you can feel free to create any other set of features (unigram, bigram, trigram, etc).
□	With the features you have created, select any traditional (non-neural network) model to train a model and provide plots on the evaluation.
□	Explain all of your choices. Note: the model performance will not necessarily be judged, however the choice of which model and why it was selected, based on the features generated will need to be explained.

TASK 3 - *Pre-trained word embeddings + linear classifier model*

□	Select any pre-trained word embedding of your choice (i.e. Glove).
□	You will run your text inputs through the word embedding to map each input instance (sequence of words) to a *single* embedding that represents the entire sequence of words. It is up to you to decide how you want to do this. The only constraint is that if your word embedding is W-dimensional, then each of your text inputs will now be W-dimensional, and hence the dataset inputs should be able to fit into a matrix of shape (N,W), where N is the dataset size, and W is the dimension of the word embedding.

□	Explain all of your choices.
□	Train a simple linear model using the embeddings as your inputs using sklearn.linear_model.SGDClassifier. Use whatever number of iterations it takes to train a model in 10 minutes MAX (don't worry, we aren't looking for accuracy here).
□	Report relevant metrics.

TASK 4 - R*ecommendations to the clients*

□	Create a final plot(s) of the relevant performance metrics from each experiment.
□	Your job is to present this to each client, providing a recommendation to the clients, taking into consideration all of the clients wants and needs.
□	Explain your decisions.

TASK 5 - *Submit exam to Statistics Canada*

□	See instructions under "What to Submit" on the first page of this document. Submit your exam at statcan.fellowship-fellowship.statcan@statcan.gc.ca Include: - all code used to complete the task and other required auxiliary code - share the link to your public Git repository so we can evaluate the code, figures and documentation