**Classifying customer complaints using Watson NLP**

**Explore text classification with Watson NLP**

**Learn the fundamentals of IBM Watson NLP, and step through the process of training and evaluating the models to perform text classification**

With IBM Watson Libraries, IBM introduced a common library for artificial intelligence (AI) runtimes (for serving the model) and AI libraries (like Natural Language Processing, Document Understanding, Translation, and Trust). IBM Watson Libraries brings everything under one umbrella for consistency and ease of development and deployment. This tutorial walks you through the steps of training a model to classify text in consumer complaints by using the watson\_nlp library from IBM Watson NLP.

The watson\_NLP library is available on IBM Watson Studio as a runtime library so that you can directly use it for model training, evaluation, and prediction. The following figure shows the Watson NLP architecture.

IBM Watson NLP is a standard embeddable AI library that is designed to tie together the pieces of IBM Natural Language Processing. It provides a standard base natural language processing layer along with a single integrated roadmap, a common architecture, and a common code stack designed for widespread adoption across IBM products.

Text Classification is usually manually processed by humans to gather groups of qualitative data. Having the ability to automatically gather and process larger data sets of text through customer feedback, comments, or an entire article that is written on your product is a strong tool to gain insight into the most common emotional responses in a group of people or a block of text.

IBM Watson NLP now provides the ability to automatically classify the input text into one or more predetermined sets of labels.

This tutorial explains the fundamentals of IBM Watson NLP and walks you through the process of training and evaluating the models to perform text classification.

Prerequisites

To follow the steps in this tutorial, you need:

* An [IBMid](https://cloud.ibm.com/login?cm_sp=ibmdev-_-developer-tutorials-_-cloudreg%22%20\\t%20%22_blank" \t "_blank)
* A Watson Studio project
* A Python notebook
* Your [environment set up](https://developer.ibm.com/tutorials/set-up-your-ibm-watson-libraries-environment/)

Before working through the tutorial, you should have an understanding of IBM Watson Studio and Jupiter Notebooks.

**Estimated time** :It should take you approximately 1 hour to complete this tutorial.

This notebook demonstrates how to train text classifiers using Watson NLP. The classifiers predict the product group from the text of a customer complaint. This could be used, for example to route a complaint to the appropriate staff member.

The data that is used in this notebook is taken from the Consumer Complaint Database that is published by the Consumer Financial Protection Bureau (CFPB), a U.S. government agency. The Consumer Complaint Database is a collection of complaints about consumer financial products and services that the CFPB sent to companies for response. A complaint contains the consumer’s narrative description of their experience if the consumer opted to share this information publicly and after the Bureau has removed all personal information. In this notebook, you will focus on complaints that contain this narrative description to show how to use Watson NLP.

The data is publicly available at <https://www.consumerfinance.gov/data-research/consumer-complaints/>.

What you'll learn in this notebook

Watson NLP implements state-of-the-art classification algorithms from three different families:

* Classic machine learning using SVM (Support Vector Machines): SVM is a support vector machine classifier, which may be trained using any type of input embedding / vectorization block’s predictions as feature vectors, e.g., USE embedding’s, TF-IDF vectorizers. Supports multi-class and multi-label text classification and produces confidence scores via Platt Scaling.
* Deep learning using CNN (Convolutional Neural Networks): CNN is a simple convolutional network architecture, built for multi-class and multi-label text classification on short texts. Utilizes GloVe embeddings.
* BERT: BERT is a transformer-based architecture, built for multi-class and multi-label text classification on short texts. Utilizes Multilingual BERT pretrained models.
* A transformer-based architecture: Transformer is a transformer-based architecture, built for multi-class and multi-label text classification on short texts. Utilizes BERT and RoBERTa pretrained models.

Watson NLP also offers an easy to use *ensemble classifier* which combines different classification algorithms and a majority voting.

In this notebook, you'll learn how to:

* Prepare your data so that it can be used as training data for the Watson NLP classification algorithms.
* Train a TF-IDF SVM model using watson\_nlp.workflows.classification.TFidfSvm. TFidfSvm workflow simplifies the training process of TF-IDF vectorization+SVM
* Train a USE SVM model using watson\_nlp.workflows.classification.UseSvm. UseSvm workflow simplifies the training process of USE embedding+SVM
* Train a Glove CNN model using watson\_nlp.workflows.classification.GloveCNN. GloveCNN workflow simplifies the training process of Glove embedding + CNN
* Train a VotingEnsemble using watson\_nlp.workflows.classification.GenericEnsemble. The GenericEnsemble model combines three classification models: CNN with Glove Embedding, SVM with TF-IDF and SVM with USE (Universal Sentence Encoder). It computes the weighted mean of classification predictions using confidence scores.
* Store and load classification models as an asset of a Watson Studio project.
* Score data and compare model quality by running the models on test data and using the built-in quality evaluation and building a custom confusion matrix.

Table of Contents

1. Before you start
2. [Data Loading](https://github.com/ibm-build-lab/Watson-NLP/blob/b584595c10279b3ece907e5f872fecf9c745832a/ML/Text-Classification/#loadData)
3. [Data Processing](https://github.com/ibm-build-lab/Watson-NLP/blob/b584595c10279b3ece907e5f872fecf9c745832a/ML/Text-Classification/#prepareData)
   1. [Prepare training and test data](https://github.com/ibm-build-lab/Watson-NLP/blob/b584595c10279b3ece907e5f872fecf9c745832a/ML/Text-Classification/#prepareTraining)
4. [Model Building](https://github.com/ibm-build-lab/Watson-NLP/blob/b584595c10279b3ece907e5f872fecf9c745832a/ML/Text-Classification/#buildModel)
   1. [Train a TF-IDF SVM classification model with Watson NLP](https://github.com/ibm-build-lab/Watson-NLP/blob/b584595c10279b3ece907e5f872fecf9c745832a/ML/Text-Classification/#tfidfsvm)
   2. [Train a USE SVM classification model with Watson NLP](https://github.com/ibm-build-lab/Watson-NLP/blob/b584595c10279b3ece907e5f872fecf9c745832a/ML/Text-Classification/#usesvm)
   3. [Train a Golve CNN classification model with Watson NLP](https://github.com/ibm-build-lab/Watson-NLP/blob/b584595c10279b3ece907e5f872fecf9c745832a/ML/Text-Classification/#cnn)
   4. [Train an ensemble classification model with Watson NLP](https://github.com/ibm-build-lab/Watson-NLP/blob/b584595c10279b3ece907e5f872fecf9c745832a/ML/Text-Classification/#ensemble)
   5. [Store and load classification models](https://github.com/ibm-build-lab/Watson-NLP/blob/b584595c10279b3ece907e5f872fecf9c745832a/ML/Text-Classification/#storeLoad)
5. [Model Evaluation](https://github.com/ibm-build-lab/Watson-NLP/blob/b584595c10279b3ece907e5f872fecf9c745832a/ML/Text-Classification/#evaluate)
6. [Summary](https://github.com/ibm-build-lab/Watson-NLP/blob/b584595c10279b3ece907e5f872fecf9c745832a/ML/Text-Classification/#summary)

Before you start

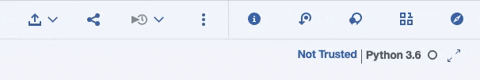
Stop kernel of other notebooks.

Note: If you have other notebooks currently running with the *DO + NLP Runtime XX.x on Python 3.x* environment, stop their kernels before running this notebook. All these notebooks share the same runtime environment, and if they are running in parallel, you may encounter memory issues. To stop the kernel of another notebook, open that notebook, and select *File > Stop Kernel*.

Set Project token.

Before you can begin working on this notebook in Watson Studio in Cloud Pak for Data as a Service, you need to ensure that the project token is set so that you can access the project assets via the notebook.

When this notebook is added to the project, a project access token should be inserted at the top of the notebook in a code cell. If you do not see the cell above, add the token to the notebook by clicking More > Insert project token from the notebook action bar. By running the inserted hidden code cell, a project object is created that you can use to access project resources.

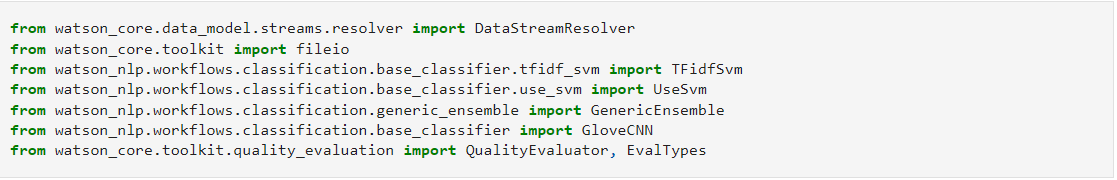


Tip: Cell execution

Note that you can step through the notebook execution cell by cell, by selecting Shift-Enter. Or you can execute the entire notebook by selecting Cell -> Run All from the menu.

Begin by importing and initializing some helper libs that are used throughout the notebook.





Data Loading (customer complaint data)

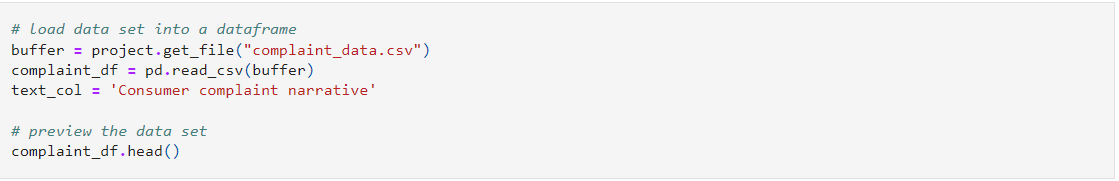
The data can be downloaded via an API from <https://www.consumerfinance.gov/data-research/consumer-complaints/>. The data is exported in CSV format.

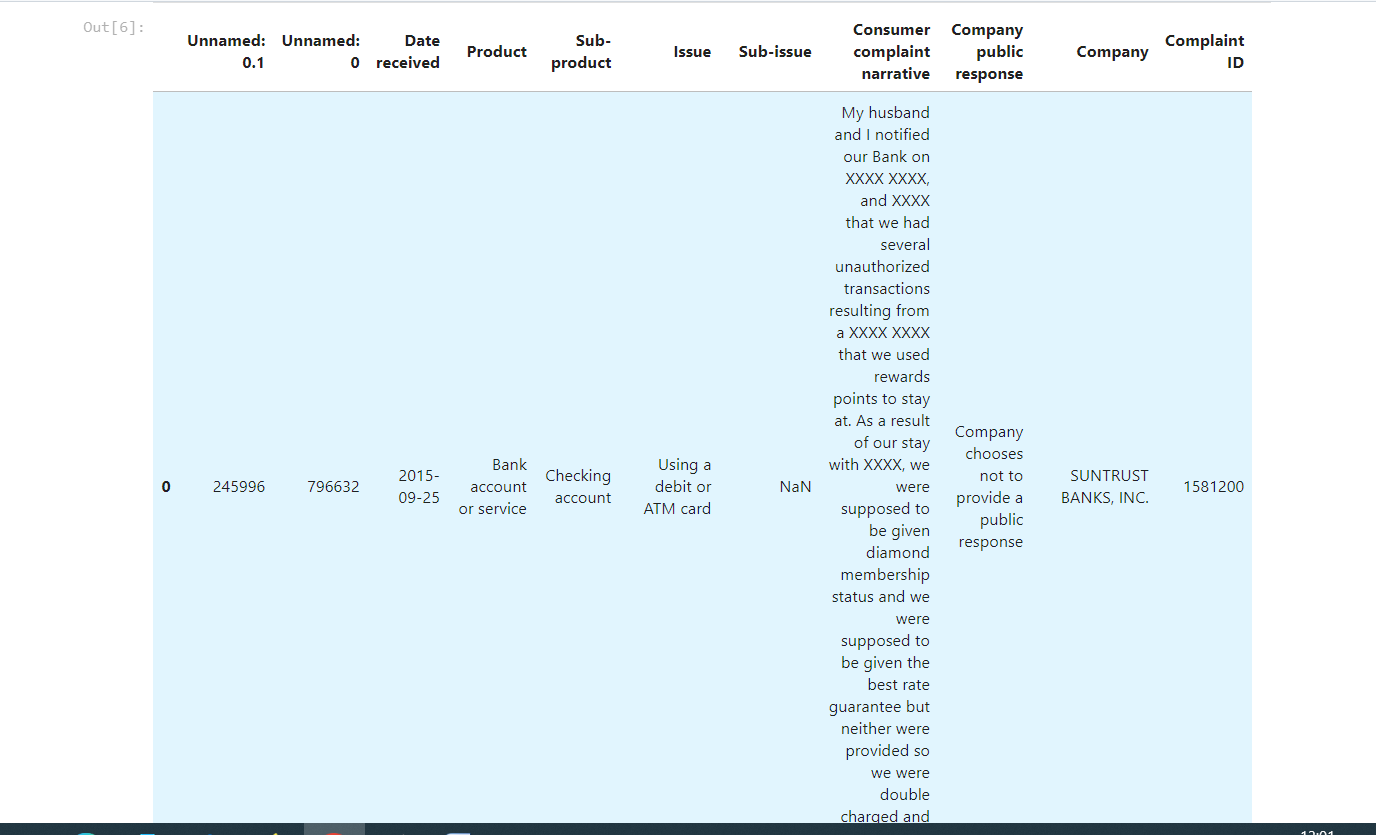
Read the data into a dataframe. You can find a detailed explanation of the available columns here: <https://www.consumerfinance.gov/complaint/data-use/#:~:text=Types%20of%20complaint%20data%20we%20publish> .

In your analysis you will focus on the *Product* column, which contains the product group, and the column with the complaint text *Consumer complaint narrative*.

We load the consumer complaints into a DataFrame.

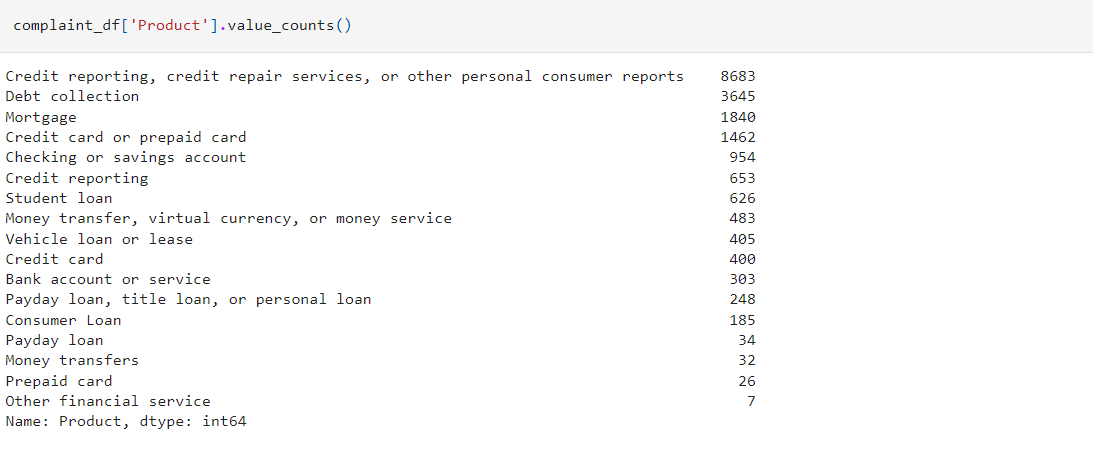
This data set contains 999285 consumer complaints with the date received, product, sub-product, submitted via and company information.





## 3. Data Processing & EDA

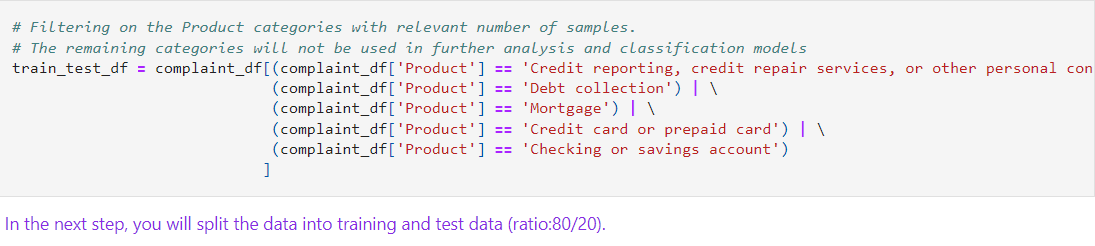
Let's look at all product groups that are available in the data set because these are the classes that the classifier should predict from a given complaint text.

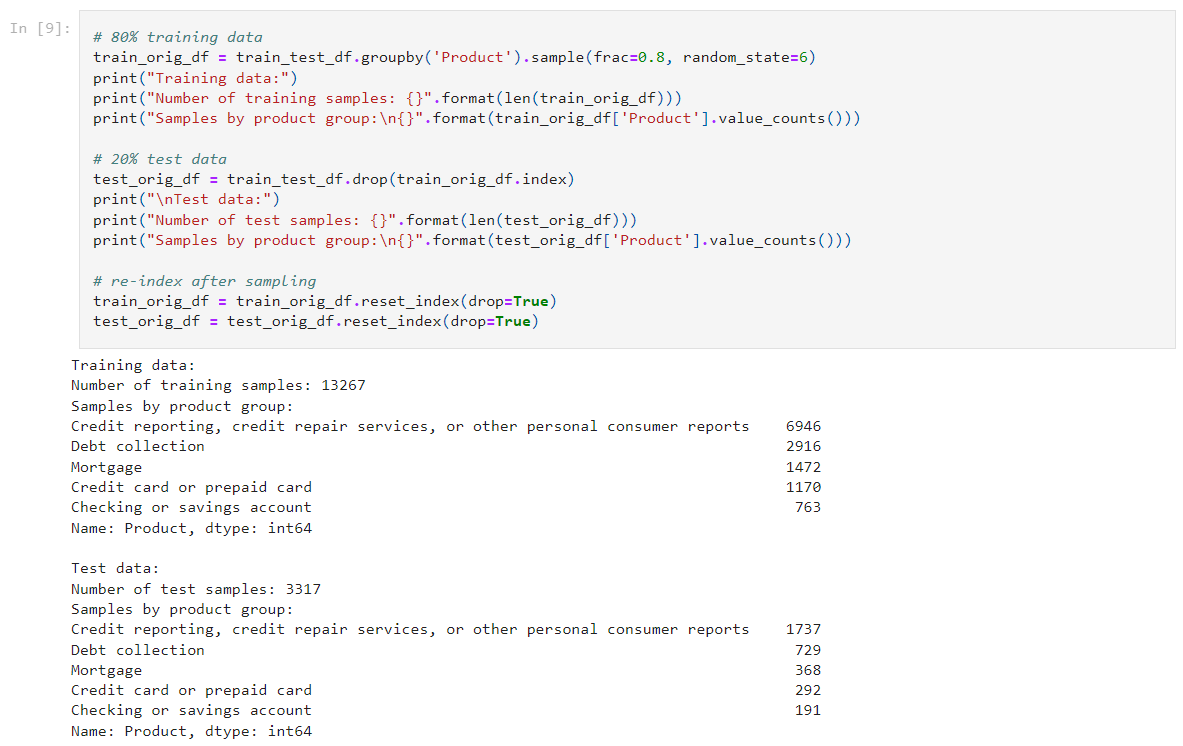


## 3.A Prepare training and test data

Many classification algorithms work best if the training samples are equally split across the classes. If the data is unbalanced, algorithms might decide to favor classes with many samples to achieve an overall good result. To avoid this, you will sample the data in the next step to have a similar amount of samples for each class.

To avoid long runtimes in this sample notebook, you will use only a small number of samples. However, this can reduce the quality of the classification models. In a real-case scenario, you should increase the number of samples per product group to get better results.

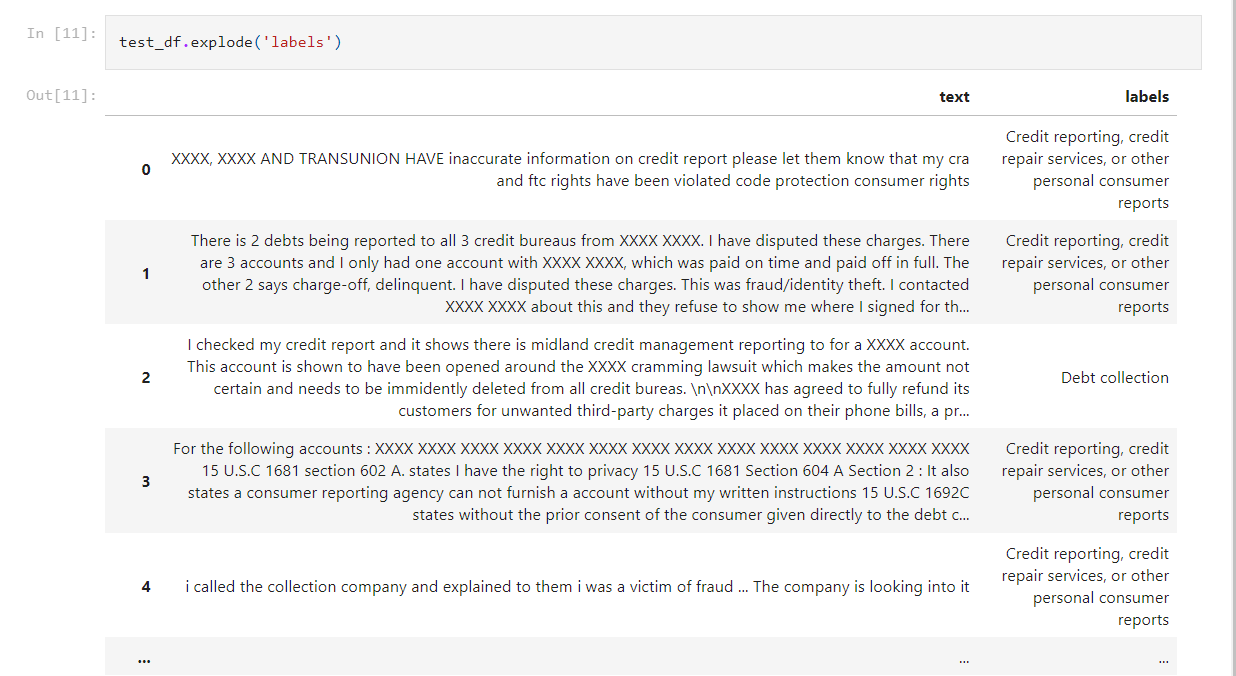


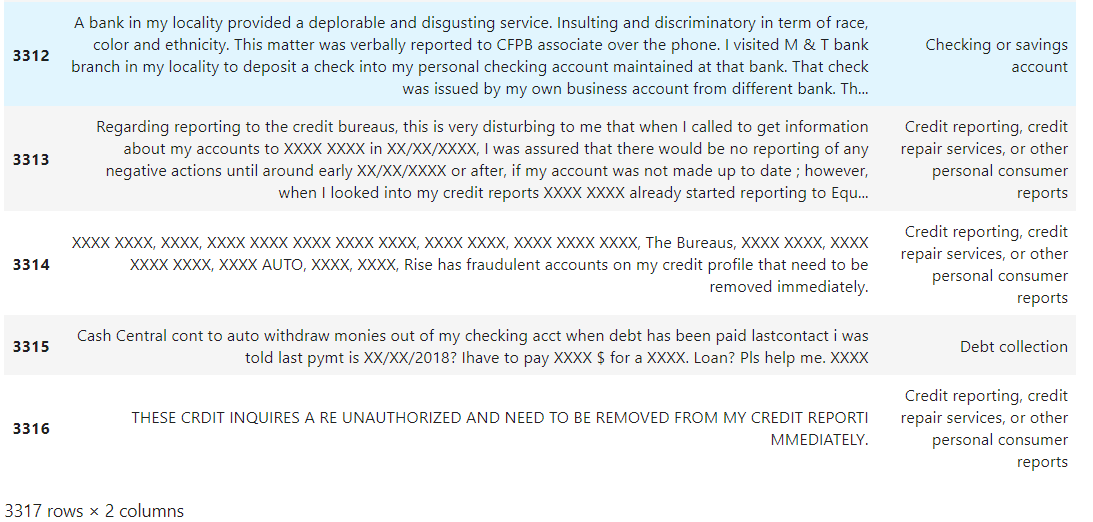


You have created two dataframes, one for the training and one for the test data. The data is still in its original format. Now you need to bring the data into a format that is usable by the Watson NLP classification algorithms. This can be either JSON or CSV format.

In the sample, you will create the data in JSON format. The training and test data is written to files.



V 



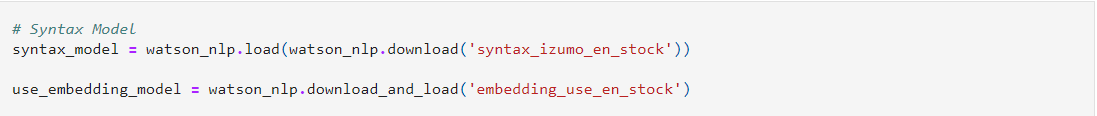


## 4. Model Building

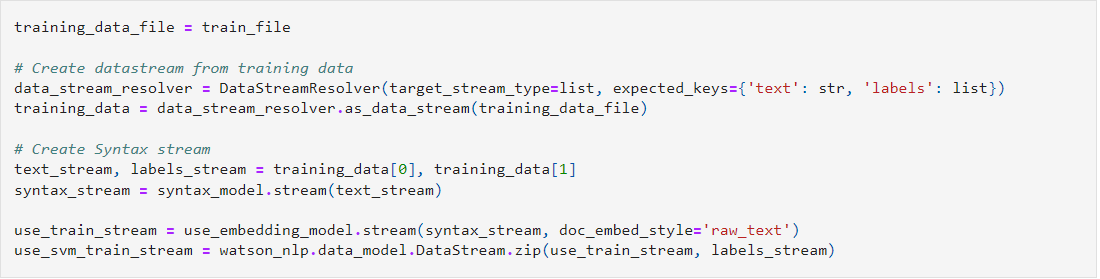
## 4.A Train a TF-IDF SVM classification model with Watson NLP

SVM is an established classification approach. Watson NLP includes an SVM algorithm that exploits the SnapML libraries for faster training. The algorithm utilizes TF-IDF embeddings.

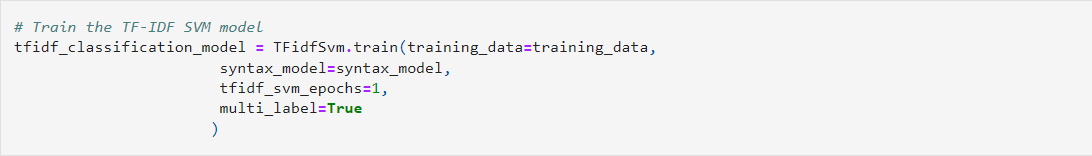
The TF-IDF SVM classifier workflow depends on the syntax block. So, start by loading the syntax model.



Classification blocks expect the training data in data streams. You can create data streams using several utility methods, as shown below.

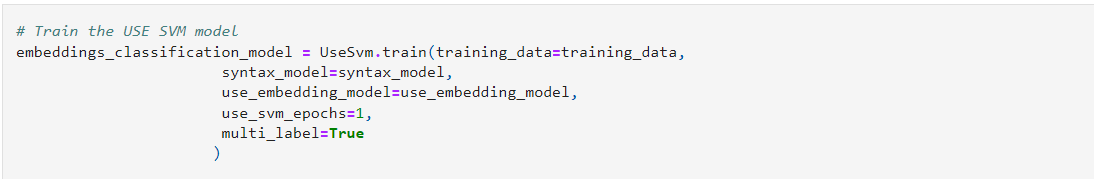


Train the classifier. **Note:** This cell will run for several minutes.



## 4.B Train a USE SVM classification model with Watson NLP

This algorithm utilizes Universal Sentence Encoder (USE) embeddings that encode word-level semantics into a vector space.



4.C Train a Glove CNN model with Watson NLP

## 4.D Train an ensemble classification model with Watson NLP

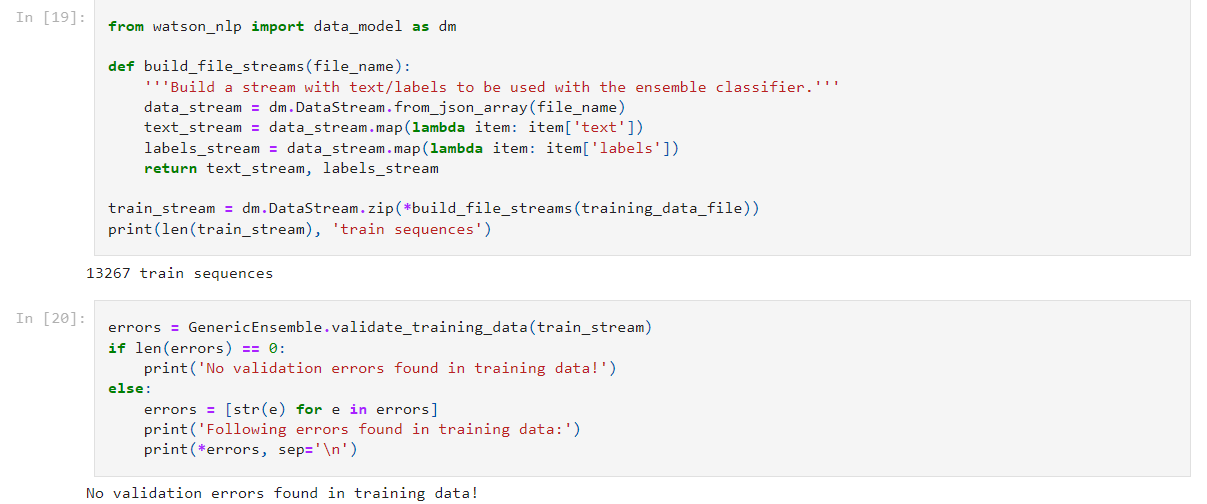
The ensemble model combines three classification models:

* CNN with GloVE embeddings
* SVM with TF-IDF features
* SVM with USE (Universal Sentence Encoder) features

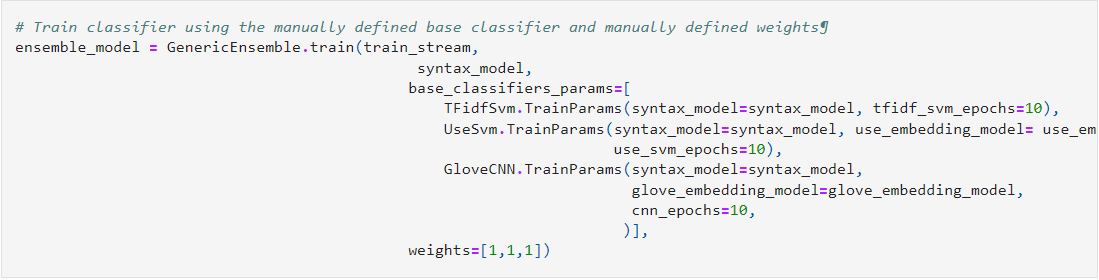
It computes the weighted mean of classification predictions using confidence scores. You will use the default weights which can be fine-tuned in subsequent steps.

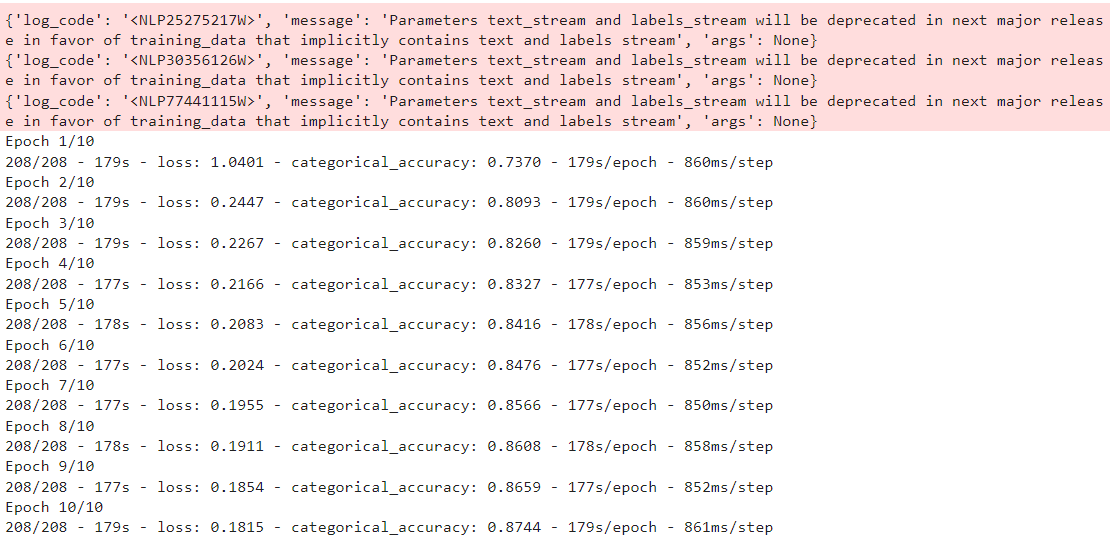
The ensemble workflow is very easy to use and the model performance can be a lot better than individual algorithms.

It depends on the syntax model and the GloVe and USE embeddings. They are passed with the file containing the training data.



Train the ensemble classifier. **Note:** This cell will run for several minutes. To restrict the time, we limited the epochs to train the classifiers to 10. This is an optional attribute - if not specified, the default will be 30 epochs.



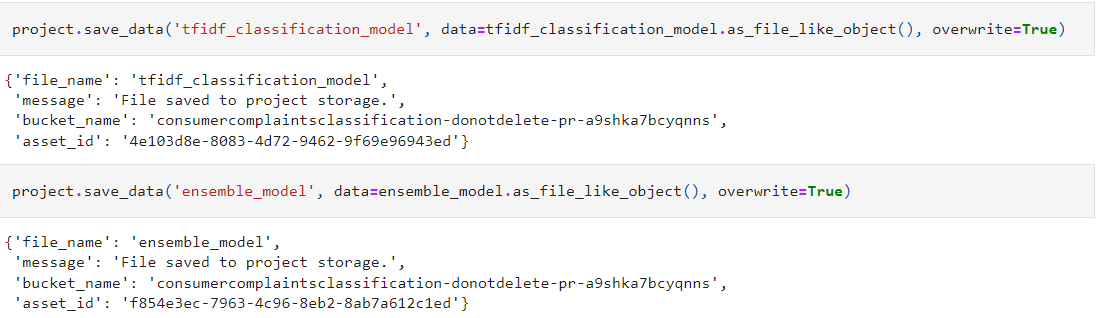


## 4.C Store and load classification models (optional)

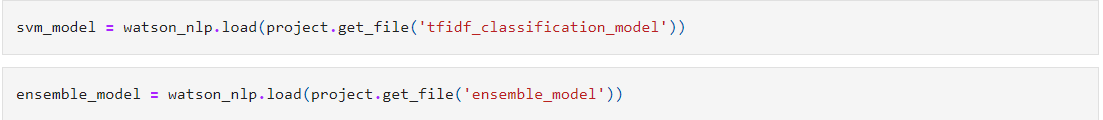
You can save a model as a project asset. model.as\_file\_like\_object() creates a ZIP archive, which is provided as a BytesIO object that is stored in the project.

**Note:** These steps are **optional**. You can skip them, and continue at [Model Evaluation](https://github.com/ibm-build-lab/Watson-NLP/blob/b584595c10279b3ece907e5f872fecf9c745832a/ML/Text-Classification/#evaluate)

Save both models in your project.



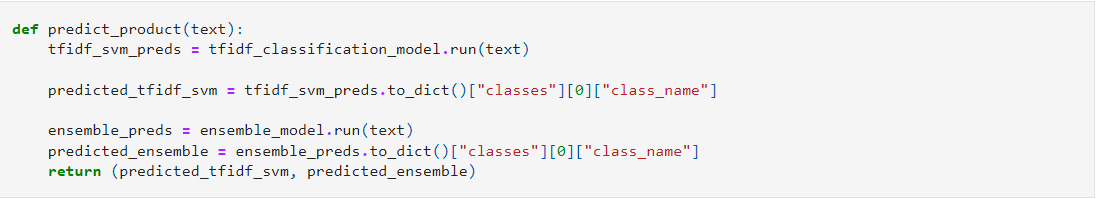
The ZIP archive created by the save\_data function is compatible to the watson\_nlp.load() function that is also used to load the predefined Watson NLP models.



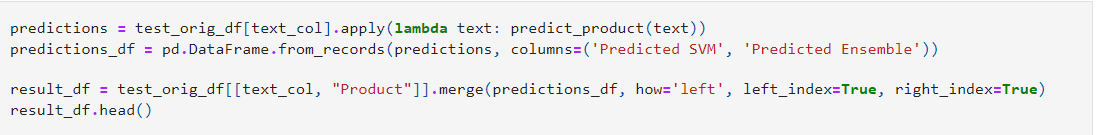
## 5. Model Evaluation

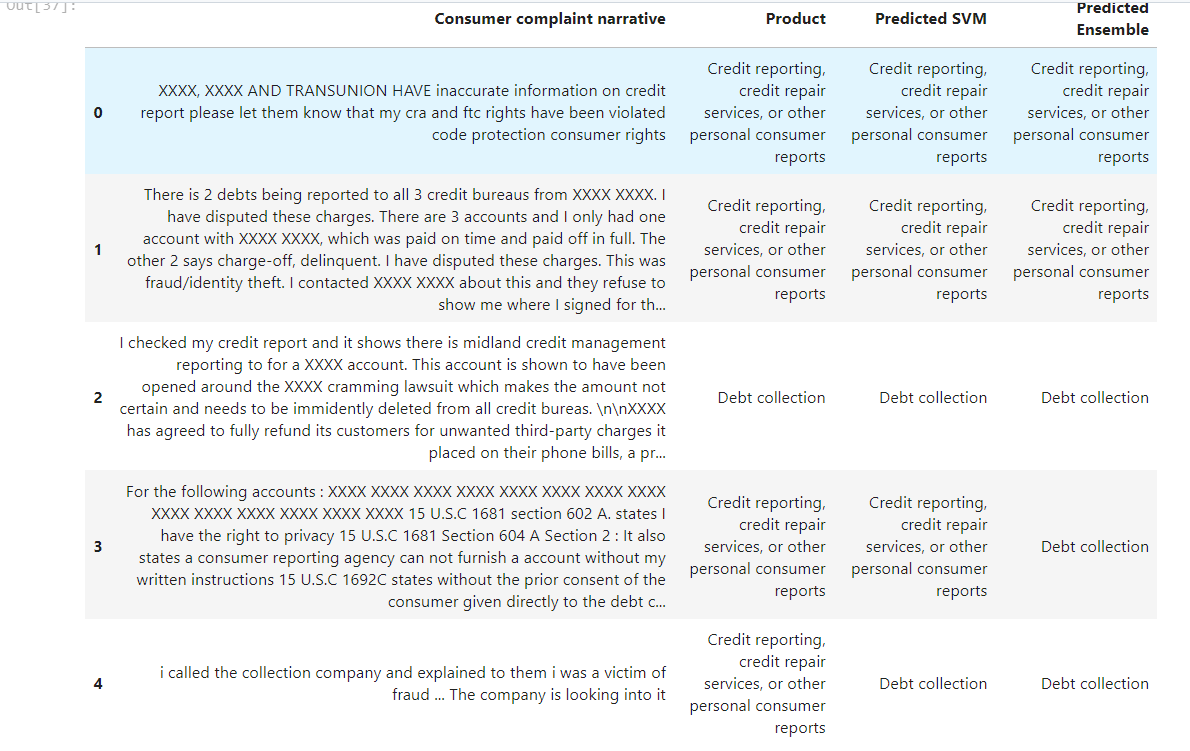
Now you are able to run the trained models on new data. You will run the models on the test data so that the results can also be used for model evaluation. For illustration purposes, the data is used in the original format that you started out with because the format of the new complaints that you receive might also be in that format.

Create a helper method to run both models on a single complaint and return the predicted product groups of both models.

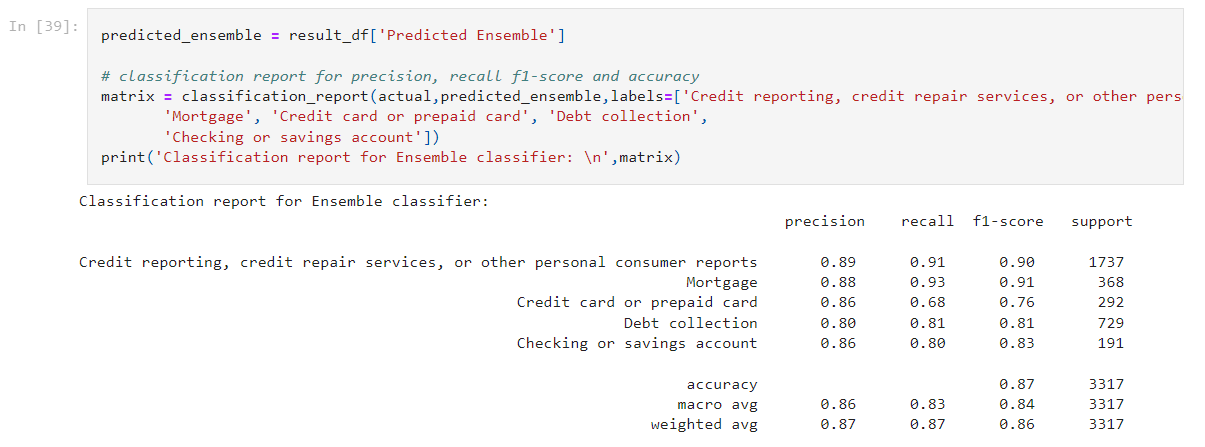


Run the models on the complete test data.





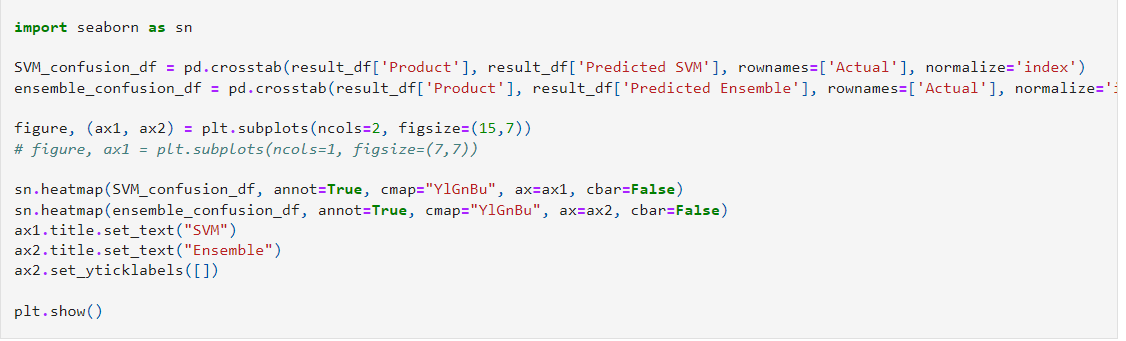
Creating and plotting a confusion matrix

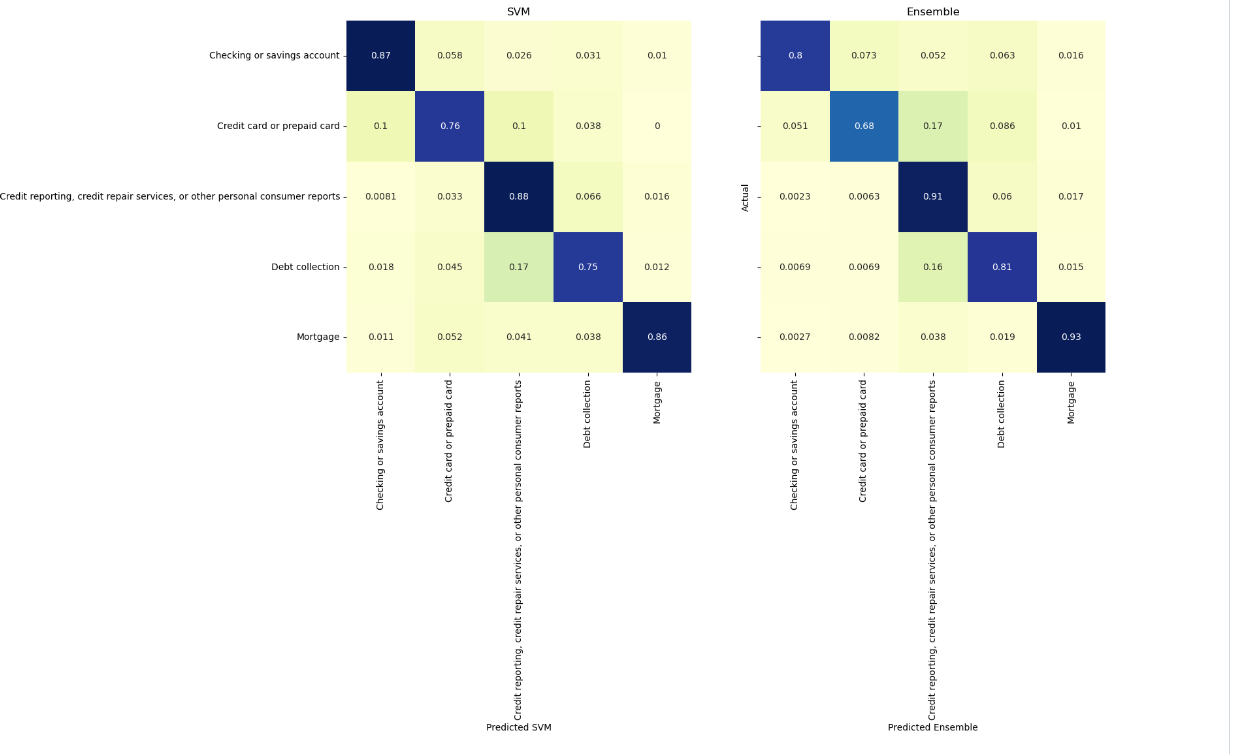


You can see that the precision, recall and f1-measure for some classes is much lower than for others. The reason might be that it is difficult to differentiate between some classes.

To find out if this is true, create a custom confusion matrix to see if there are classes that seem to be very close and might have been classified inappropriately.

Use the pandas crosstab to create a confusion matrix for both the SVM and the ensemble model and plot them as Seaborn heatmaps.





**In the confusion matrix for the SVM model you can now see that complaints for Money transfer, virtual currency, or money service are often misclassified as Checking or savings account. Other common misclassifications can be gathered from the table.**

**Overall, the ensemble model performs at par with the SVM model. However, the SVM model had a significantly shorter training time.**

In subsequent steps, consider increasing the size of the training data, tuning the CNN training parameters or adjusting the weights of the ensemble model to gain better results