LIVE PROJECT - CLOUD COUNSELAGE PVT. LTD. IP PROGRAM

**CLOUD EVENTS**

RECOMMENDATION SYSTEM PROJECT REPORT

horizontal line

# Introduction

In this era, the corporate world is inculcated in extensive programmes for their employees to have the potential to promote creativity and productivity. The task of recommending every eligible employee in the company regarding an event that is to be organized is sometimes waste of resources and time. To improve this task we build a model to recommend company events which is more likely to be preferred by a certain employee. One often misses events of interest sheerly due the lack of awareness at the right time. Cloud Counselage also receives invites for events in multiple domains that need to be forwarded to the people with relevant interests. This was mentioned in the problem statement for the live project for the Event Recommendation System. So we came up with a simple event recommendation system for the employees of the company, ‘“*Cloud Events*’’.

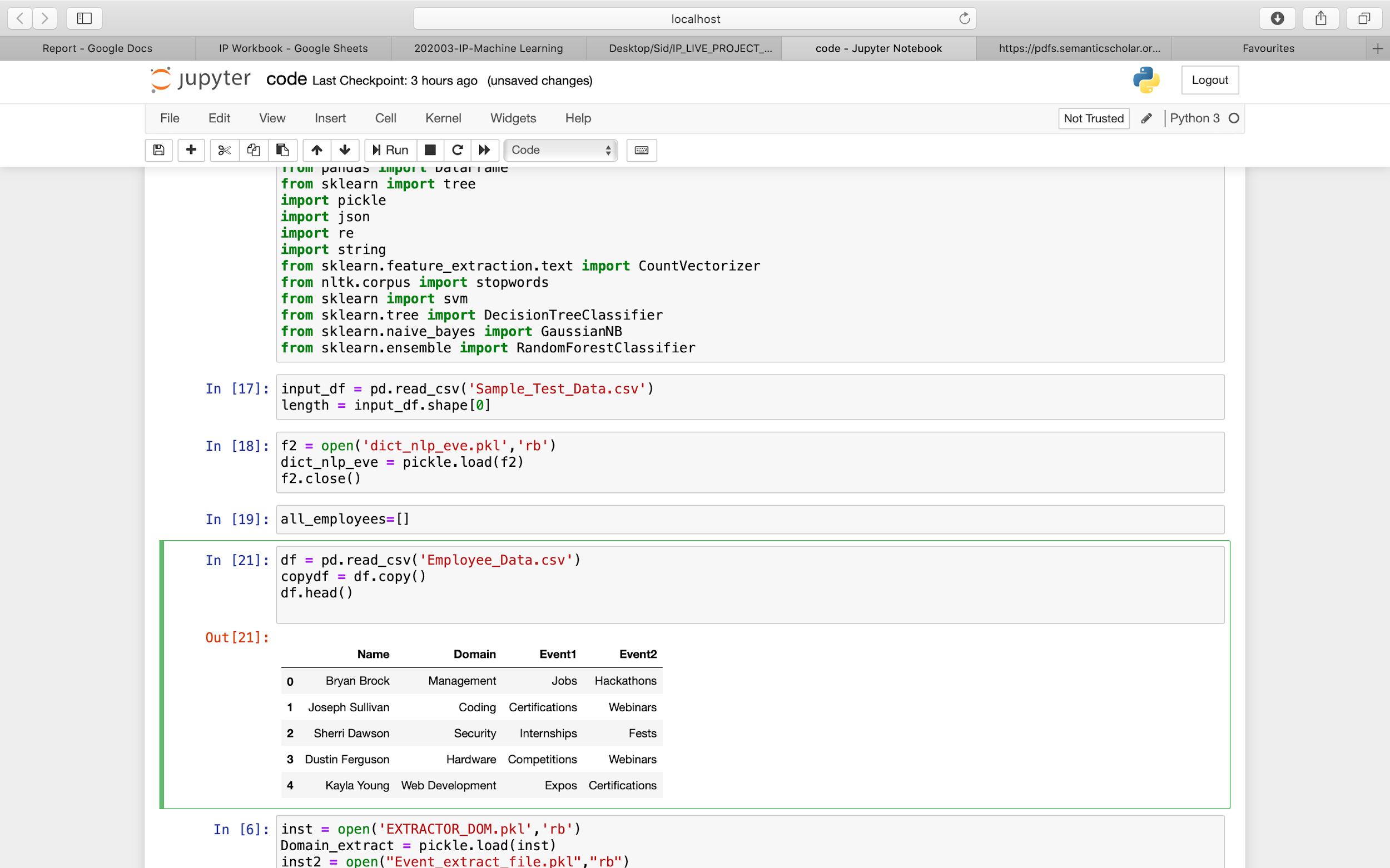
Cloud Events is a reliable program that uses a Machine Learning approach that analyzes the Event name, its brief description if provided with it, and then finds items that seem relevant for company employees to be interested in. Then there is a selection of employees to whom this event is to be recommended and then they are displayed in an excel sheet as an output.

# Problem Setting

Goal is to create a Recommendation System based on employee domains and event type preferences provided to us in the CSV document. The system should read the new events and autonomously classify them into various domains, then for each event the system should output a list of employees whose preference matches with the event's detected domain.

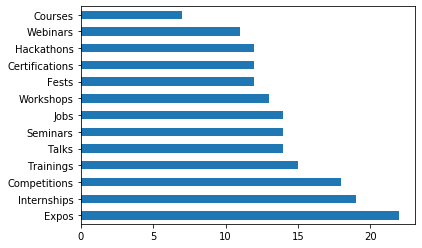
# Dataset

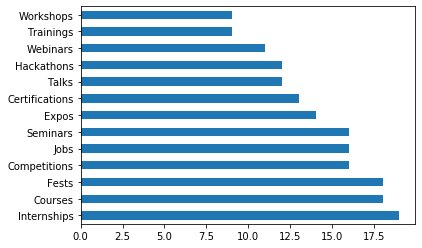
The CSV document has 4 columns and 183 rows.

The dataset has 183 employees, 23 unique Domains and 13 unique event types.

There is a Name column consisting of employees' names, and the domain column which consists of the domain they are affiliated to.

For event types they have two preferences namely columns Event1 and Event2. There is no priority assigned to them i.e. both event columns have to be taken into consideration equally.

Below are some bar graphs to represent the layout and some special characteristics of our dataset:

Event [ 1 ] Event [ 2 ] 

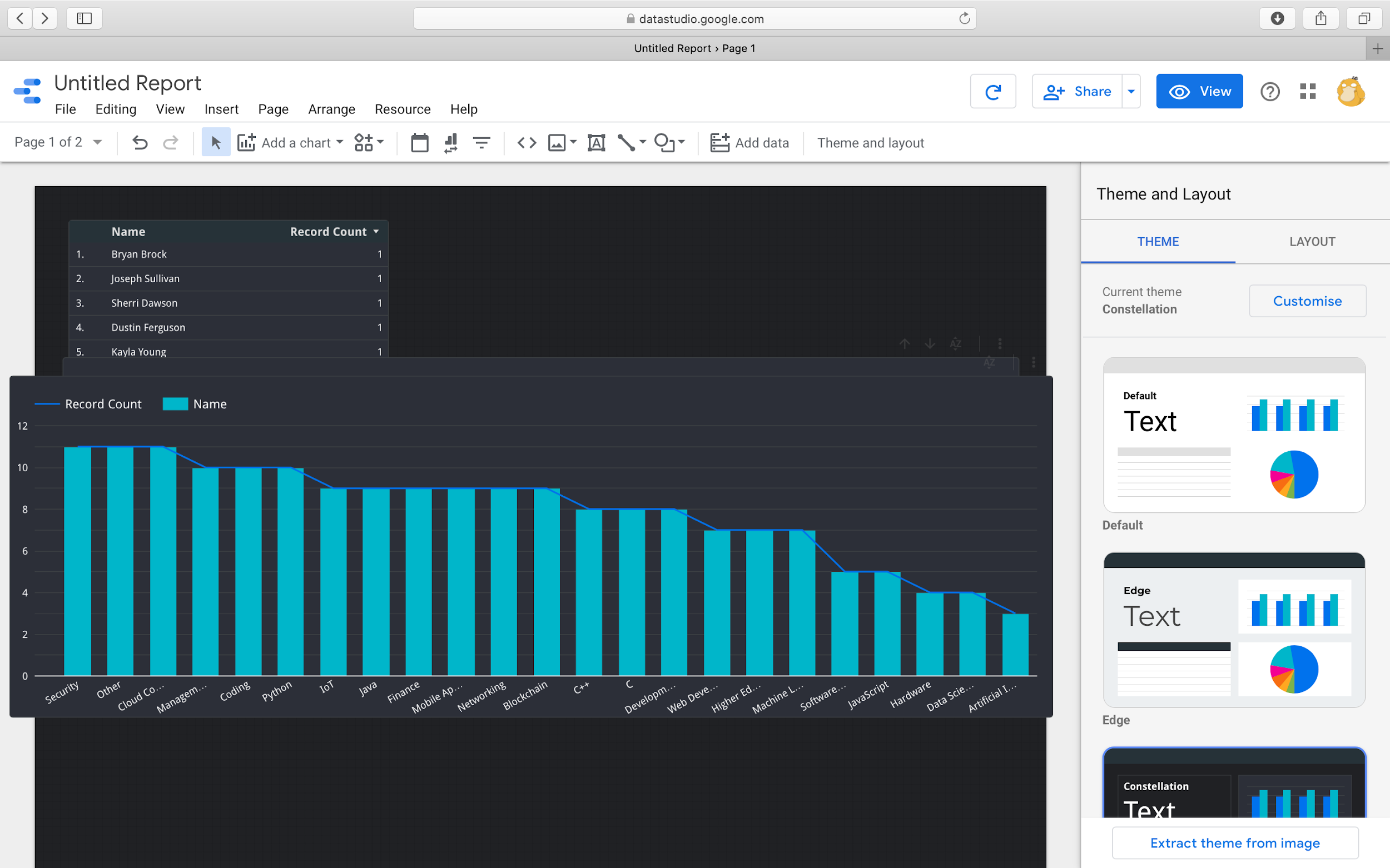


Figure Event [1] and Event [2] are two bar graphs representing unique event counts. Within all the domains, internships are really preferred event types with the highest sum count in both columns.

Above figure shows the domain which employees are affiliated to, with Security, Cloud Computing, and Others being the highest and domains related to Machine Learning, AI, Data Science being the lowest.

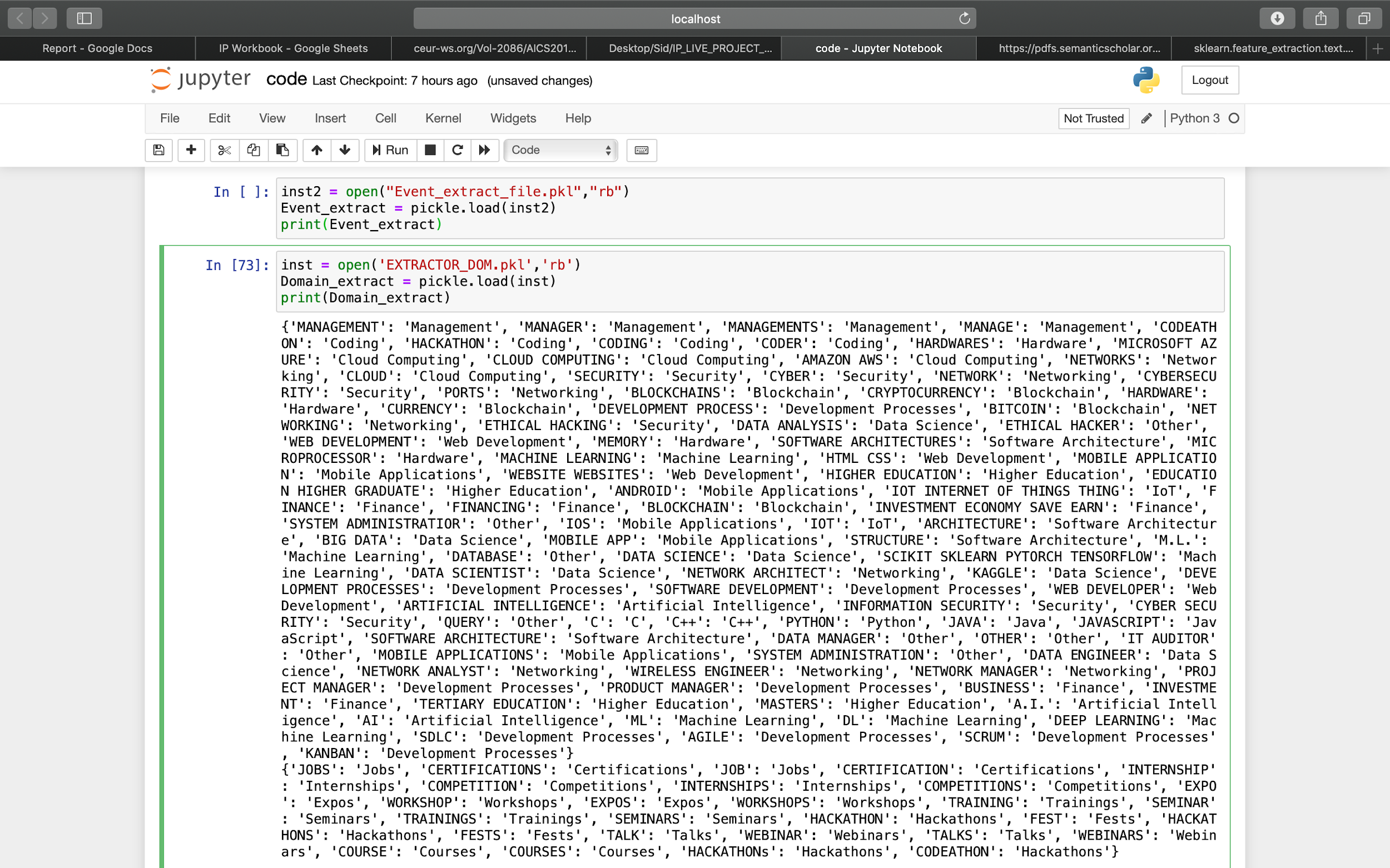
*NOTE: The problem statement refers to classification of domains and event types to get employee names. However ML approach is applied to figuring out the domain and events from the input i.e. a NLP model is used. Hence we can come to the conclusion that the employee data is to be used only to match domain and event to get employee names. Hence there is no need for EDA on the data set and we can continue with our NLP problem.*

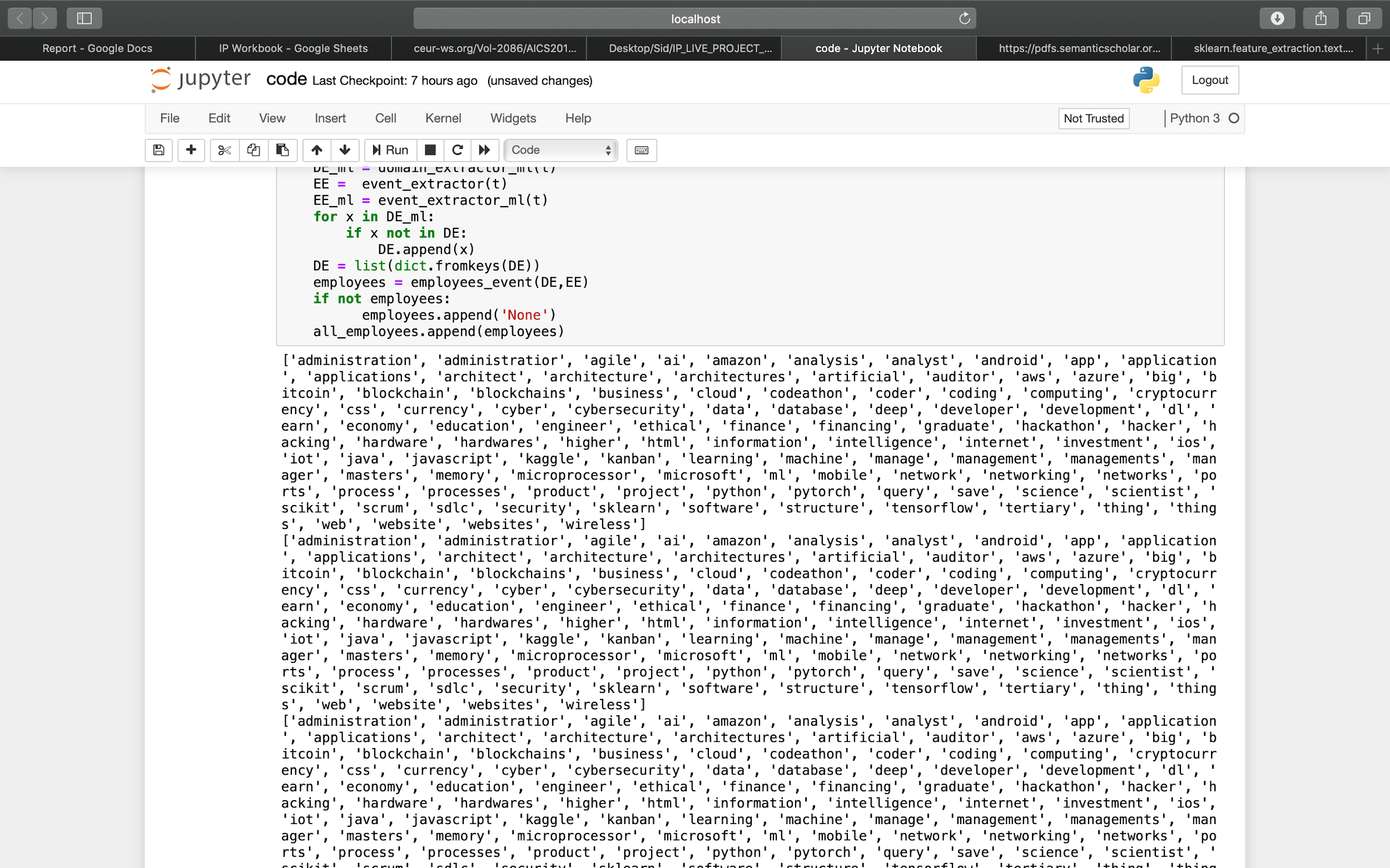
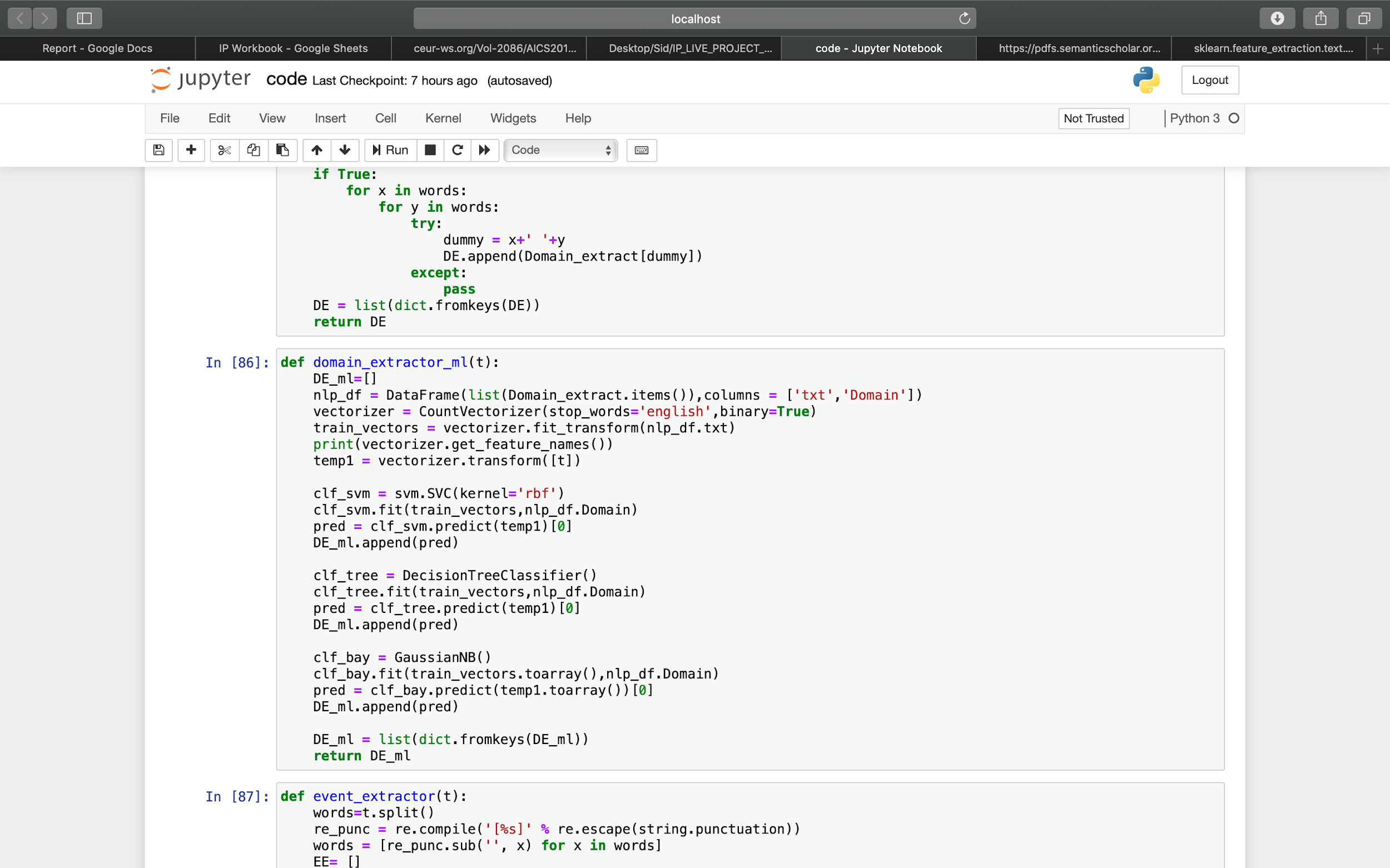
# Domain and Event BoW model:

The input fed to be fed to our model is a CSV where each instance is an Event name with domain and event affiliated. Here we used the BoW model to deal with the problem.

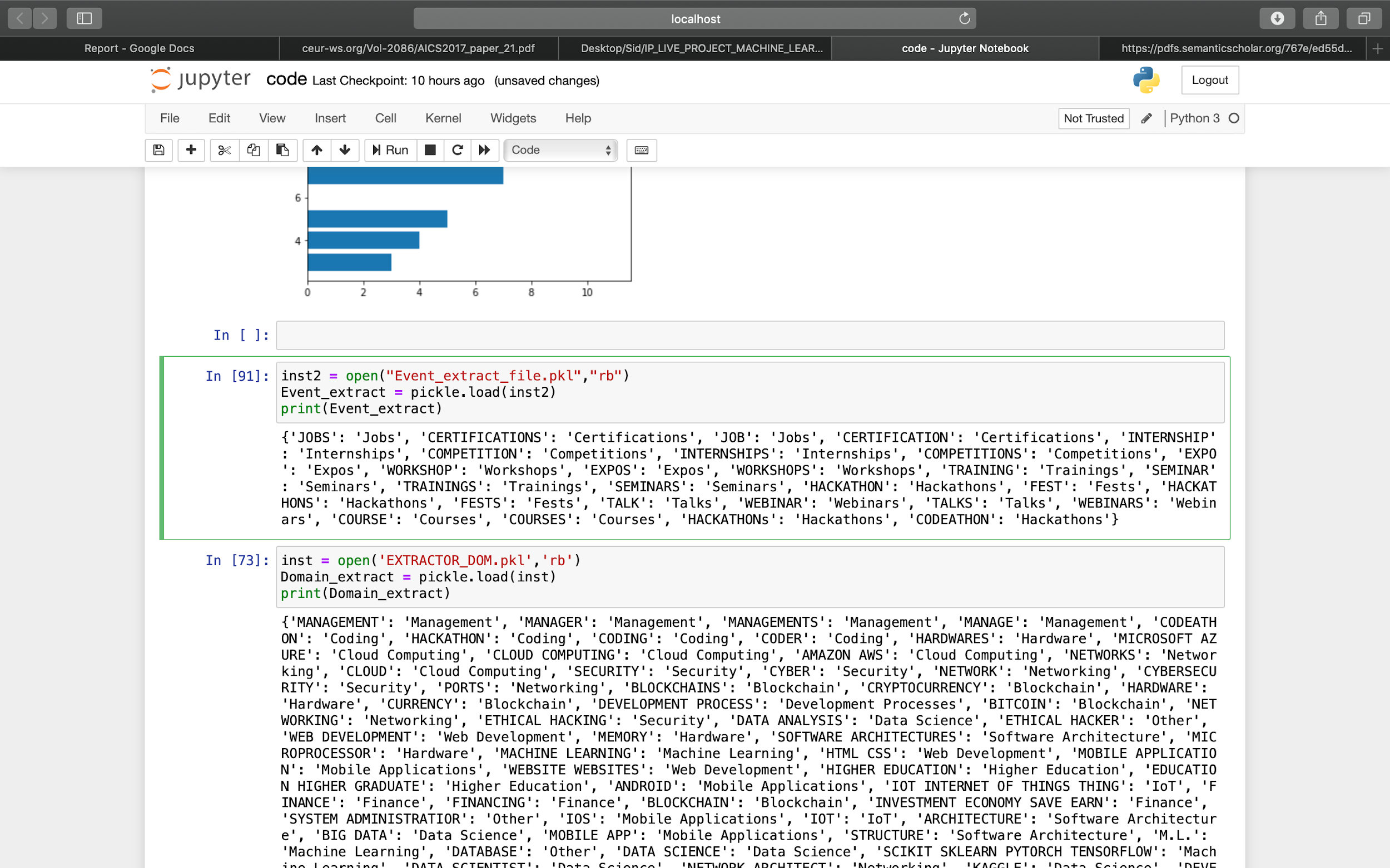
Bag-of-Words or BoW is a simplifying representation used in [natural language processing](https://en.wikipedia.org/wiki/Natural_language_processing) and [information retrieval](https://en.wikipedia.org/wiki/Information_retrieval) (IR). In this model a text is represented as the [bag](https://en.wikipedia.org/wiki/Multiset) of words, disregarding grammar and even word order but keeping [multiplicity](https://en.wikipedia.org/wiki/Multiplicity_(mathematics)). Bag-of-words features are created by viewing the document as an unordered collection of words, which are then used to classify the document. However, the bag-of-words features have very little understanding of the semantics of the words, which can be measured as the distances between words in an embedding space. This is because words are treated as atomic units and therefore there is no notion of similarity between words.

For this model I had to create text by myself. *A dictionary was created to store domains and text affiliated to them*. The domains were the keys and fields were key values. Same procedure was followed for events. This dictionary was too big and not at all presentable. So to increase code readability, these two dictionaries were stored in a file using the pickle library.The bag-of-words model builds a vocabulary from all of the words in the dictionary key values text. It then models each text by counting the number of times a word in the vocabulary/tokens appears in the text. Here tokens will be generated by the words inside the key value texts stored inside the dictionaries. Below is a partial snapshot of the Domain dictionary.





Above is a snapshot of the tokens generated by using CountVectorizer for domain extraction. These tokens were filtered with the help of the “ re “ library. Filtration such as removal of stop words helped in keeping the token matrix concise and accurate. It removes common words such as ‘is’, ‘the’, ‘from’, etc. For the bag-of-words approach, the reviews were cleaned via a text-processing algorithm to remove any unwanted characters, HTML links or numbers and retrieve only raw text. The tokens are then saved in a dataframe with their respective domains e.g. Data Science - [‘Data’, ‘Science’, ‘EDA’, ‘Kaggle’, …]. This dataframe is then used in a classification algorithm which will extract the domains and events. The classification model is trained on the basis of these two features. We then use the predict function for our input events set. These input event sets are first split and tokenized using CountVectorizer before feeding them in the predict function.

Below is a partial image of the Event dictionary.

# 

# Training Machine Learning Models

Choosing the best machine learning model for our project was the top priority in the development phase. In parallel to machine learning models, direct domain and event extractor are included in the model which may find the domain affiliated in the input event string, *if available directly within the string*. Various models were tested for and were found providing really poor performance. However some capabilities which on model lacked in were coped up in another. There were many instances where the Sample\_Data\_input.csv was fed to the model and two models were needed to be coupled to increase the performance. That itself was not the case. The models were fed vectorizer matrices which contained binary values only (CountVectorizer was given a ‘binary’ parameter). Hence when the model was used for prediction by using *predict\_proba()* function only a single output is generated. The probability of the output being correct will be abnormally higher than others and hence more distinct. However when multiple domains are passed this would fail as the probabilities of the top second domain would be really low. This situation is also sort of overcomed by deploying different models at the same time. We use Decision Tree, SVC and Naive Bayes models for our project.

## Decision Tree

The decision tree model proved to be the most appropriate model for our project despite its shortcomings. It is the first base classifier model in the set. The model works on the entropy of the overall system. That it recursively builds different trees and outputs the best tree with lowest entropy levels. However with the arising problem due to its overfitting tuning the hyperparameter is evaded. The model output accuracy for Sample\_Data\_input.csv was 0.64, frankly it is not a really good model. However this was the highest accuracy we get for Sample\_Data\_input.csv.

## Support Vector Classification

SVM is used in the project for text classification as they achieve good performance on text categorization tasks compared to other models. A reason for this is that they possess the ability to generalise well in high dimensional feature spaces and eliminate the need for feature selection. The model had accuracy of 0.57 for Sample\_Data\_input.csv. Hence it was used as the second baseline.

## Naive Bayes

Naive Bayes classifier is successfully used in various applications such as spam filtering, text classification, sentiment analysis, and recommender systems. It uses Bayes theorem of probability for prediction of unknown class.

Bayes Theorem : *P(A | B) = P(B | A)* P(A) / P (B)

However as I mentioned earlier the for our dummy data Decision tree was most suitable. Though, Naive Bayes models using bag-of-words features can still achieve impressive results, making it a valid baseline classifier.

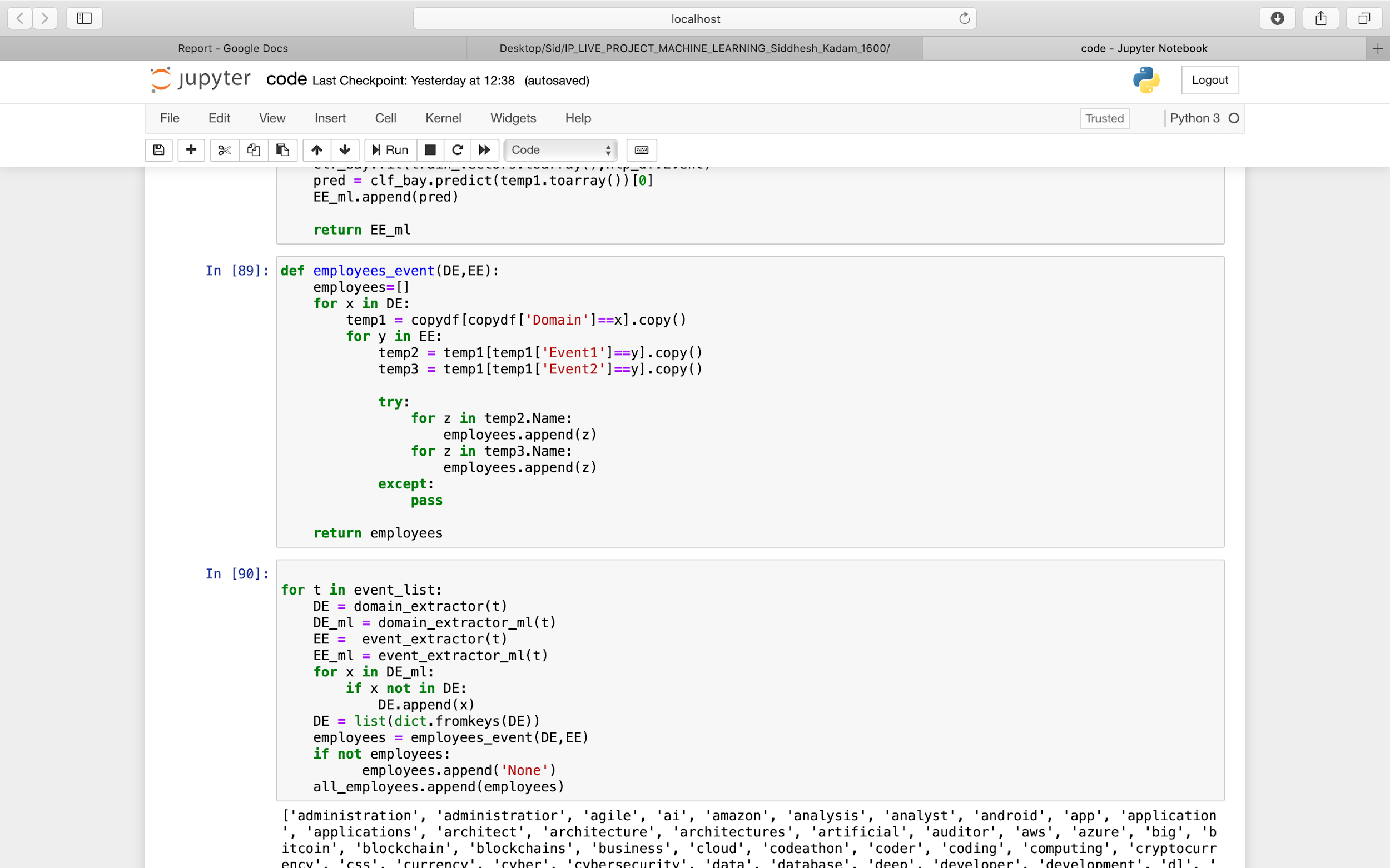
# Employee Name Search

After extraction of the domain and event, a simple string matching algorithm was implemented to fetch the employees list which have the preferred domain and event type. The search had to be implemented in the three parts:

[1]Matching the domain.

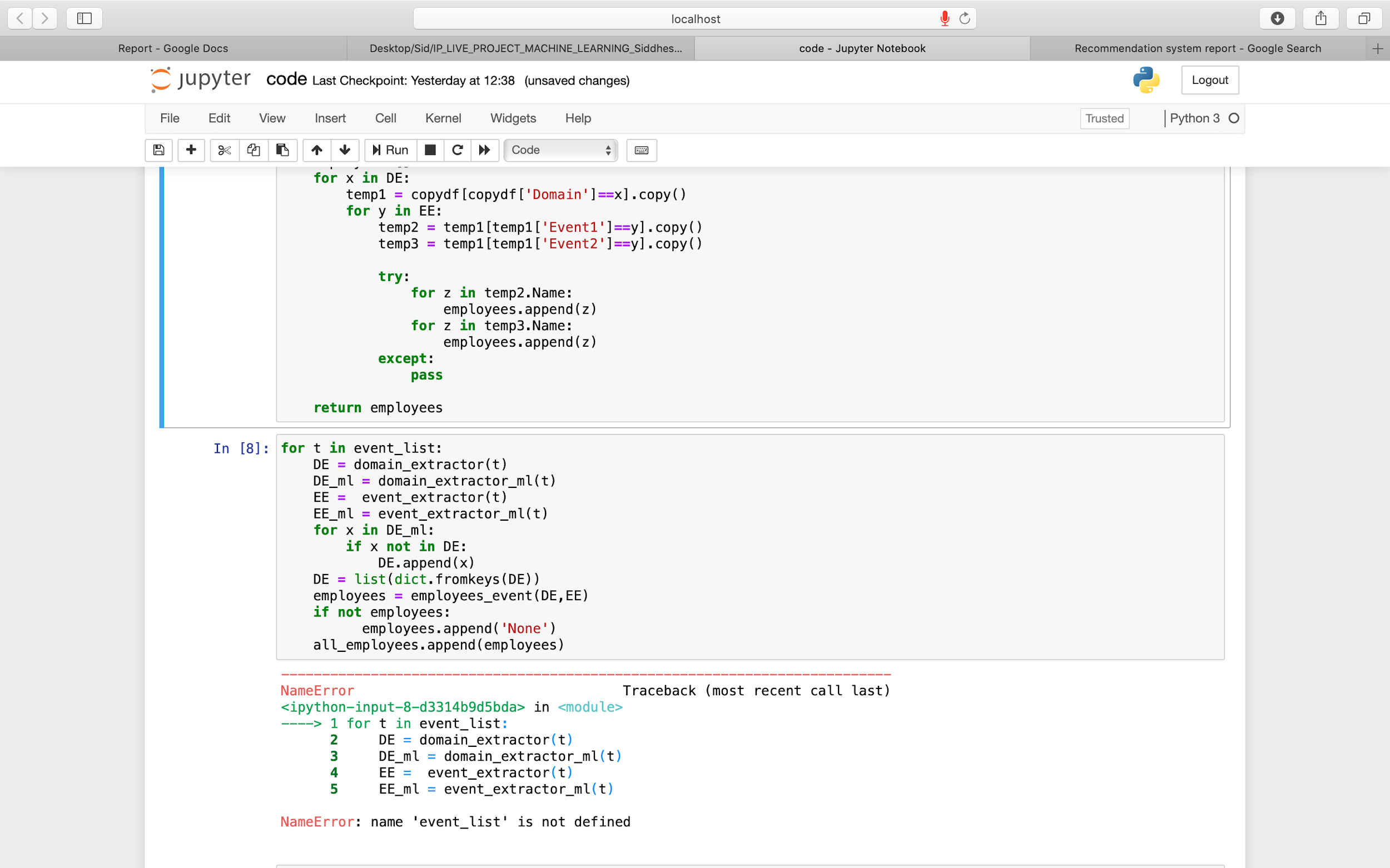
[2]Matching events with Event1.

[3]Matching events with Event2.



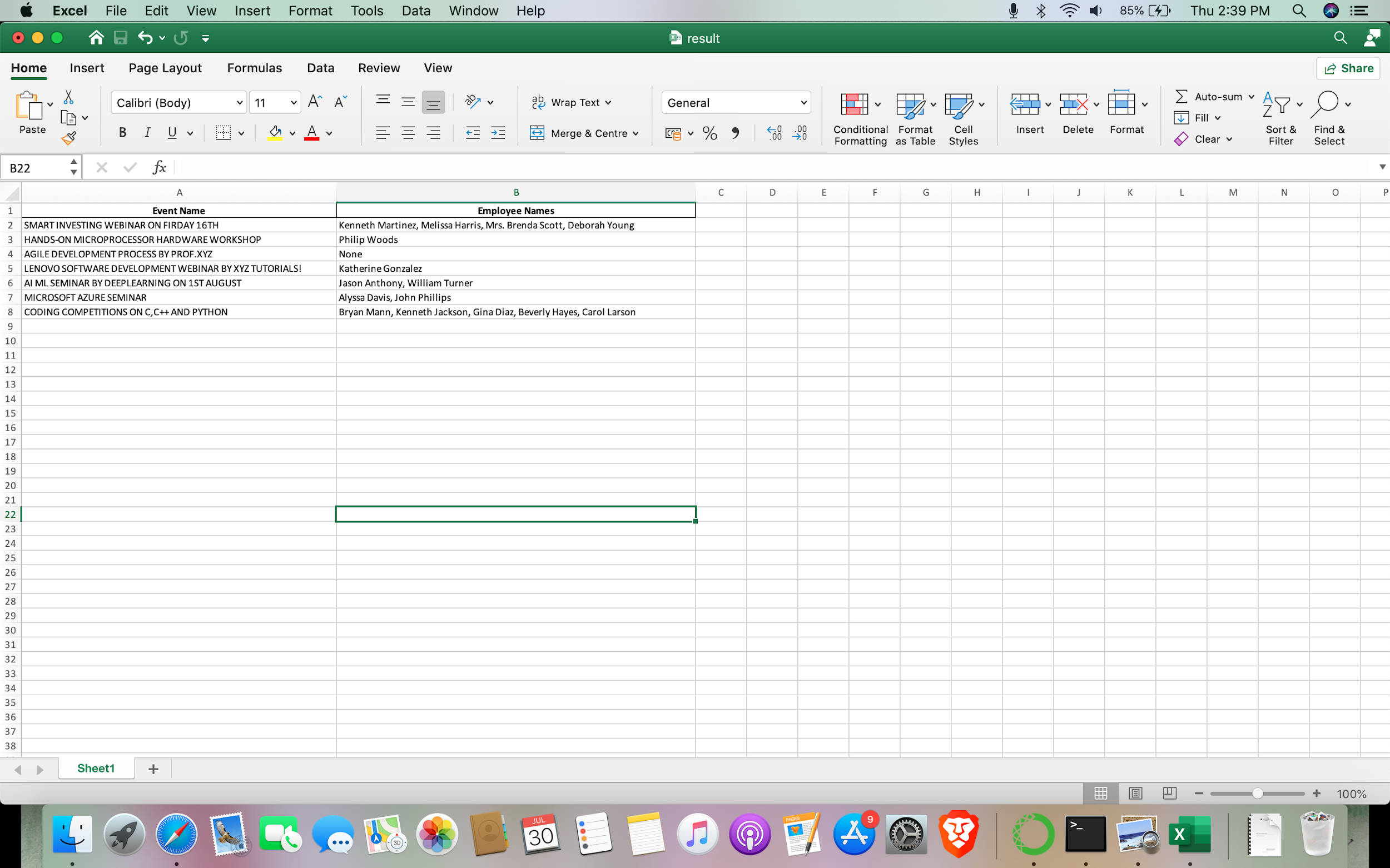
Above snapshot shows the employee\_event function that uses the string matching algorithm for domains and events for employee names with the help of pandas.

All this code however needs to be run in a loop for the total number of events in our total CSV file. Hence a for loop is initiated for “*event\_list*” and all the employees for each event instance are saved in a temporary list which is then embedded in the “*all\_employees”* list.



# Project Output

From the employee\_event function, we receive a list of employee names which the event is to be recommended to. However, this has to be displayed in a presentable manner in an excel sheet as per our problem statement. Hence we make a dataframe containing two columns namely “*Event Name*” and “*Employee Names*”.



We then append each event in the Event Name column and their respectively affiliated employee names in the other. This dataframe is then saved as an excel sheet with the help of “*to\_excel*” . A dummy sample output is saved as “*result.xlsx*” and if the program is run again the “*output.xlsx*” will be saved in the same directory as the “*code.py”* file.

# Conclusion

A hybrid approach is taken to implement the Cloud Events recommendation system. This approach overcomes drawbacks of each individual algorithm and improves the performance of the system. Techniques like SVC, Naive Bayes and Decision Trees are used to get better recommendations and increase precision and accuracy. In future we can work on hybrid recommender using a better text document for our vectorizer for better performance.

Tokens generated by our texts are weak for any project bigger than this. Our approach can be further extended to next domains such as using an API or chatbot to send recommendations.