Tentative Payment Date Predictor

Submitted in partial fulfillment of the requirements for the award of degree of

**BACHELOR OF ENGINEERING IN**

**COMPUTER SCIENCE & ENGINEERING**



**Submitted to: Submitted By:**

**SHIVAM-18BCS3054**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

# Chandigarh University, Gharuan

**2022**

**CERTIFICATE**

This is to certify that the work embodied in this Project Report entitled **“ Tentative Payment Date Predictor”** being submitted by **SHIVAM -18BCS3054** of 8th Semester for partial full-fillment of the requirement for the degree of **“ Bachelor of Engineering in Computer Science & Engineering ”** discipline in “ **Chandigarh University** ” during the academic session Jan-Jun 2022 is a record of bonafide piece of work, carried out by student under my supervision and guidance in the **“ Department of Computer Science & Engineering ”, Chandigarh University.**

**DECLARATION**

I, student of Bachelor of Engineering in Computer Science & Engineering, 8th Semester, session:, Chandigarh University, hereby declare that the work presented in this Project Report entitled “Tentative Payment Date Predictor” is the outcome of my own work, is bonafide and correct to the best of my knowledge and this work has been carried out taking care of Engineering Ethics. The work presented does not infringe any patented work and has not been submitted to any other university or anywhere else for the award of any degree or any professional diploma.

**ABSTRACT**

Keeping a steady cash flow is one of the biggest if not the biggest problem that Small to Medium Enterprises (SMEs) deal with daily. Within the different types of cash flow, Accounts Receivable (AR) classifies the balance of money that needs to be paid by the company’s customers. In the most typical case, after receiving goods or services, the customer receives an invoice with the amount that is owed to the supplier. However, this often does not happen before the aforementioned date, meaning that the invoice is often paid late. Intervention requires resources and over-intervention could cause unwanted customer dissatisfaction. Knowing whether an invoice is going to be paid late can be vital information. Current methods of late payment prediction focus only on the history between the seller and the buyer and are unusable when this history is not present. Intuitively, one’s business depends on the relationships and transactions that it has with its neighbors. Suggesting that neighbor behavior could be useful when predicting the cash flow of a company. Unfortunately, this type of information is not always given and needs to be data mining from non-relational data. This work presents a method for building a relational network of SMEs using entity resolution and improving the current state of the art of late payment prediction using features extracted from the graph.

**TABLE OF CONTENT**

1. Introduction
2. Project Design
3. Coding
4. Tools used
5. Results

# Introduction

The tentative payment date predictor is a B2B project which tends to predict payment dates from companies using previous data sets. This project uses machine learning models to construct logic on the data that is available from the previous records and construct a model which uses data for prediction. A B2B model is a type of business where a company doesn't deal with the real customers directly but it provides services to the companies and exchange business with them. This type of business run on a credit system which means that goods are given to the companies and shops on a postpaid basis and the amount of time a shop or company takes to pay the money in a given span of time affects their credit score in the market and their cap of product withdraw from the market. We have made this project for the companies giving products on credit that so that they can have safe play with companies that would tend to pay their loan on time and the companies who won't and this would help company in doing fair business with minimal loss.

We use an ensemble of classiﬁers and achieve a precision of 89.3% and recall of 82.7% on the paid late invoices. The key contributions of our research are :

1. Using machine learning to predict the payment status of invoices to minimize the penalties incurred due to the invoices being delayed. Hence, our predictions can enable the process owners to pro-actively work on ﬂagged invoices rather than rely on teams monitoring invoice processes or conduct time consuming analysis on each invoice.
2. Modeling categorical features in a domain which has historical(temporal) in- formation as numerical features. This reduces the feature space considerably(from

∼1900 to 88) in a way that all the unique values in a category are replaced by few extra columns for each row. Otherwise, if one hot-encoding or indexing is done on categorical features, it would result in ∼1885 features.

1. We propose and evaluate an ensemble approach for invoice late payment prediction, encompassing supervised learning algorithms like Random forest and Boosted Trees which are better suited for categorical features and SVM and Logistic Classiﬁcation better suited for numerical data

# Project Design

As explained before, our problem was defined as a binary classification problem to predict if either an invoice will be payed on time or late. Although we stated the problem as predicting classes, a wide range of models return probabilities instead of just labels. This is crucial in order to do a prioritization list and rank customers with higher chances of default. Also, since the model was planned to be deployed in a client that lately will need to retrain and update the model, it is important that we use a powerful model in terms of scalability, handle missing values, and would be easy do understand the results and retrain so the non-machine learning experts could have a sense about what is going on with the data

We tested our data with five different classification methods: Naive Bayes, Logistic Regression, k-Nearest Neighbors, Random Forest [4], Gradient Boosted Decision Trees [8]. Most of features came from historical data, for

example, sum amount late invoices, total invoices late and so on. In order to calculate these features for an invoice, we needed to define a period of time that we will consider to look back. This period is different from our trained dataset that defines which invoices we will consider. To define the best range of time to look back to calculate the features, we created a parameter that we call window size. In short, window size will be the number of months prior to an invoice that we will

consider to calculate our features values. But, why not.

## Jupyter Notebook

Is a project and community whose goal is to "develop open-source software, open- standards, and services for interactive computing across dozens of programming languages". It was spun off from IPython in 2014 by Fernando Pérez and Brian Granger. Project Jupyter's name is a reference to the three core programming languages supported by Jupyter, which are Julia, Python and R, and also a homage to Galileo's notebooks recording the discovery of the moons of Jupiter.

Jupyter Notebook can connect to many kernels to allow programming in different languages. A Jupyter kernel is a program responsible for handling various types of requests (code execution, code completions, inspection), and providing a reply.

Kernels talk to the other components of Jupyter using ZeroMQ, and thus can be on the same or remote machines. Unlike many other Notebook-like interfaces, in Jupyter, kernels are not aware that they are attached to a specific document, and can be connected to many clients at once. Usually kernels allow execution of only a single language, but there are a couple of exceptions

## Tensorflow

TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.

TensorFlow was developed by the Google Brain team for internal Google use in research and production. The initial version was released under the Apache License 2.0 in 2015.

TensorFlow can be used in a wide variety of programming languages, most notably Python, as well as Javascript, C++, and Java.[11] This flexibility lends itself to a range of applications in many different sectors.

## Pandas

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

It is free software released under the three-clause BSD license.[2] The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals.[3] Its name is a play on the phrase "Python data analysis" itself.[4] Wes McKinney started building what would become pandas at AQR Capital while he was a researcher there from 2007 to 2010.

## Library features

* 1. Data Frame object for data manipulation with integrated indexing.
  2. Tools for reading and writing data between in-memory data structures and different file formats.
  3. Data alignment and integrated handling of missing data.
  4. Reshaping and pivoting of data sets.
  5. Label-based slicing, fancy indexing, and subsetting of large data sets.
  6. Data structure column insertion and deletion.
  7. Group by engine allowing split-apply-combine operations on data sets.
  8. Data set merging and joining.

# Model architecture:

## Late payment prediction

Late-payment prediction models are used on a more granular level, predicting when specific payments (i.e invoices) are going to default. While credit scoring is a well-researched topic, there has been very little research done in regards to predicting late payments

## Financial transactions

Financial transactions can be modeled as a dynamic graph to analyze the interaction between different financial bodies. Work done by [33] explores different types of metrics in an economic system model as a complex network. The network explains monetary transactions between 105 clusters, each representing an economic activity standardized by the UN. The paper provides the following two contributions: A Network definition that is as follows:

* Node, is an economic activity cluster, with the node weight being the summed transactions within the cluster.
* An un-directed Edge is present when money flows between two sectors. Its weights show the summed money flow between two clusters in either direction.

## Graph analysis and feature engineering

Graph embedding methods have seen a spike in interest and application in the last couple of years. These graph embeddings make it possible to encode graphs, making it possible to use graphs as input in various machine learning algorithms. Generally, the embedding algorithms are categorized by its method:

* Matrix Factorization: the embedding is achieved by factorization of the adjacency matrix.
* Random Walk based Deep Learning: uses the Skip Gram architecture to learn effective embeddings of random walks generated from the graphs.
* Non-Random walk Deep Learning: these methods leverage network architectures such as autoencoders or graph convolution layers to embed the input

## Feature engineering

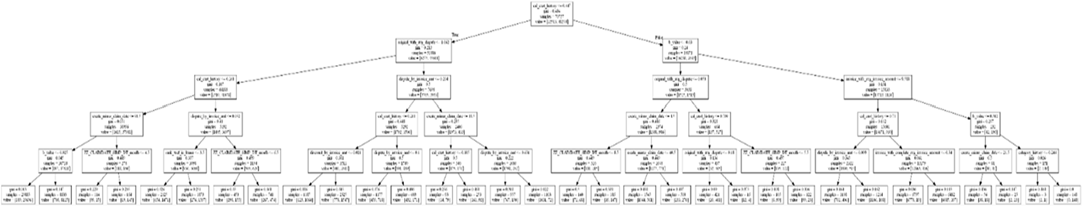
Since networks cannot directly be used as input in machine learning models. The problem of prediction relies primarily on the quality of engineered features.

Therefore, it is important to have effective techniques that extract meaningful features from the networks. A well-known problem in this domain is the problem of link prediction, where the goal is to find missing links in a network using information about the nodes. Mutluetal.

## 5. Entity Resolution

Entity Resolution (ER for short, also known as Entity Matching, Entity Disambiguation, Record Linkage) describes the problem of finding unique entities from either single or multiple data sources. The paper done by Kondaetal. describes that while there has been an effort made in understanding the problem there is very little to no published work on ER in practice, end-to-end. The general outline of the paper is to show the methodology and workflow of doing ER in a real-world scenario. It argues that every unique case needs experts to differentiate the records and heuristics. The paper contributes a description of a real-world application, the goals set by the stakeholders involved and a description of the common ER challenges in real-world applications. The authors describe the first step to be setting up the matching rules:

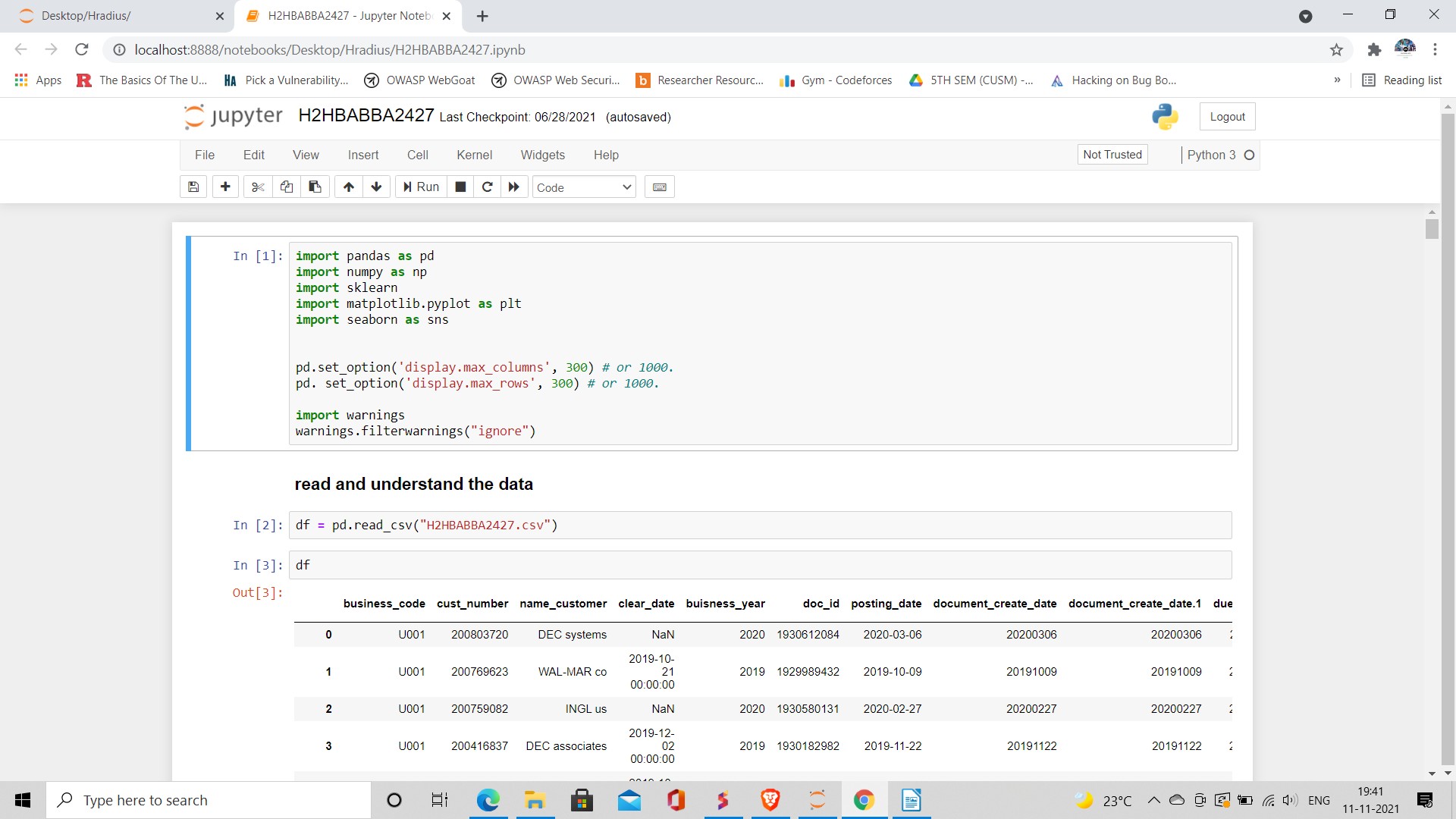
1. Two records are a direct match if the unique ID is the same in both records.
2. If Titles are similar.
3. If similar individuals are involved.



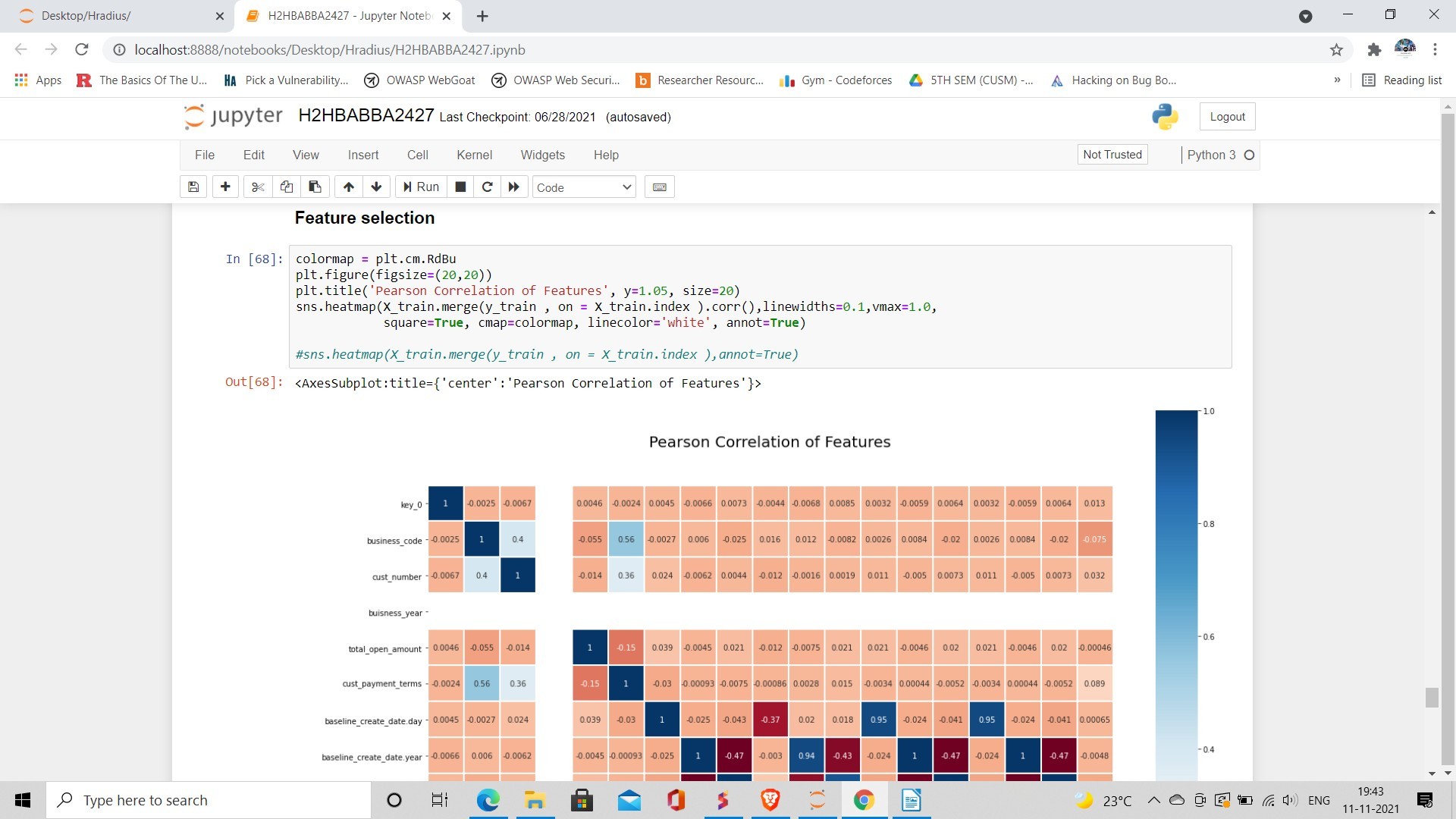
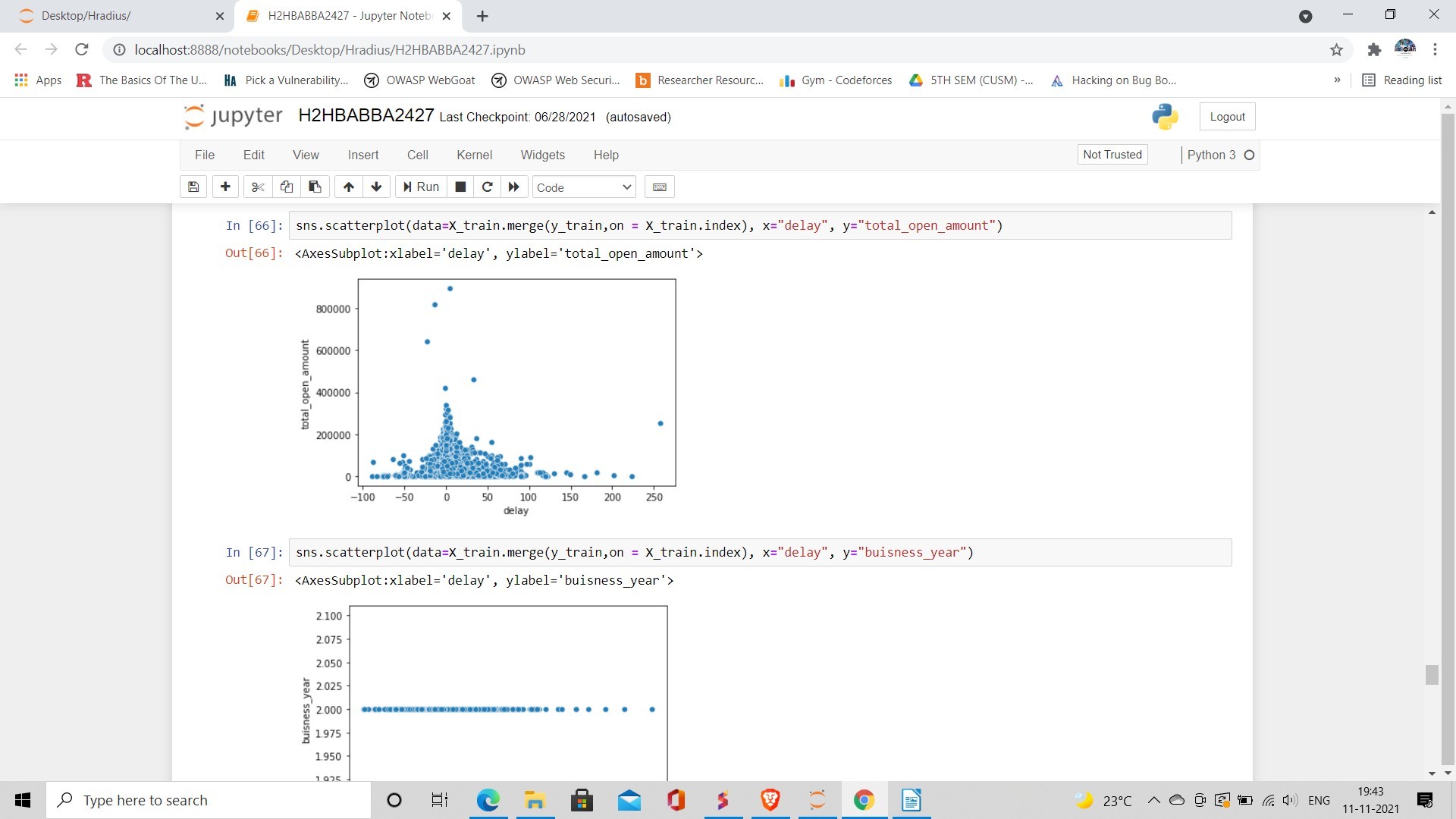
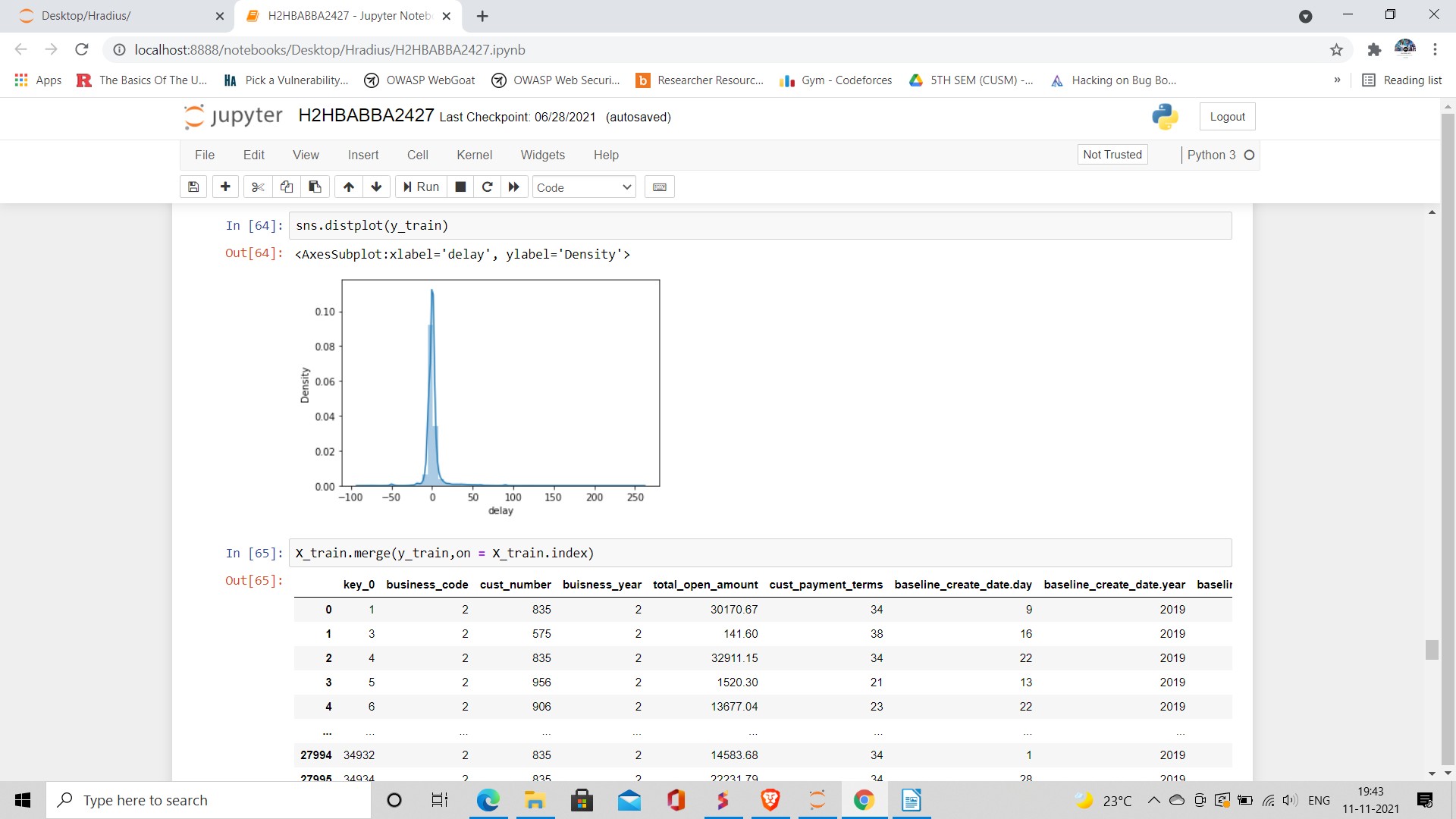
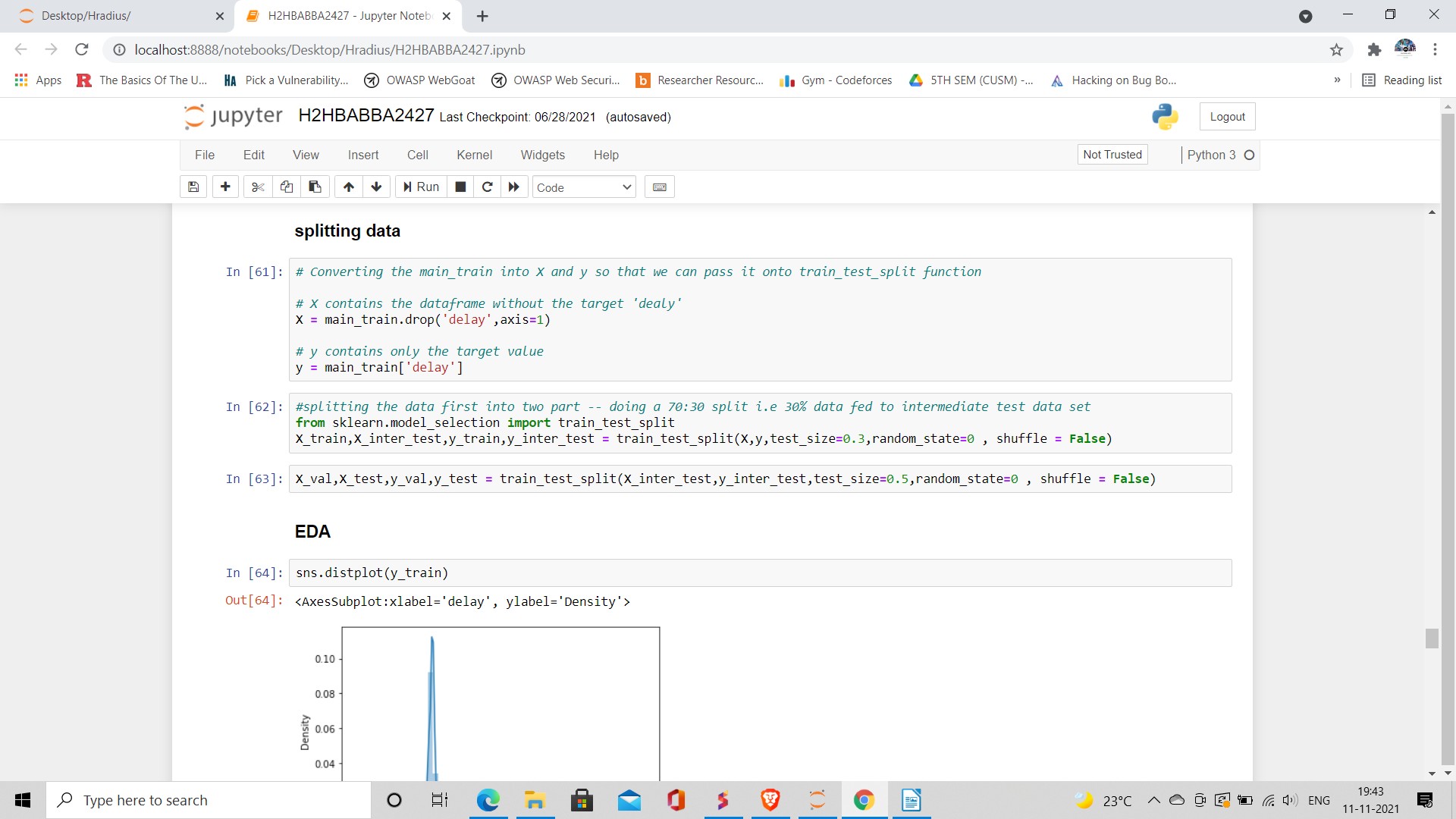
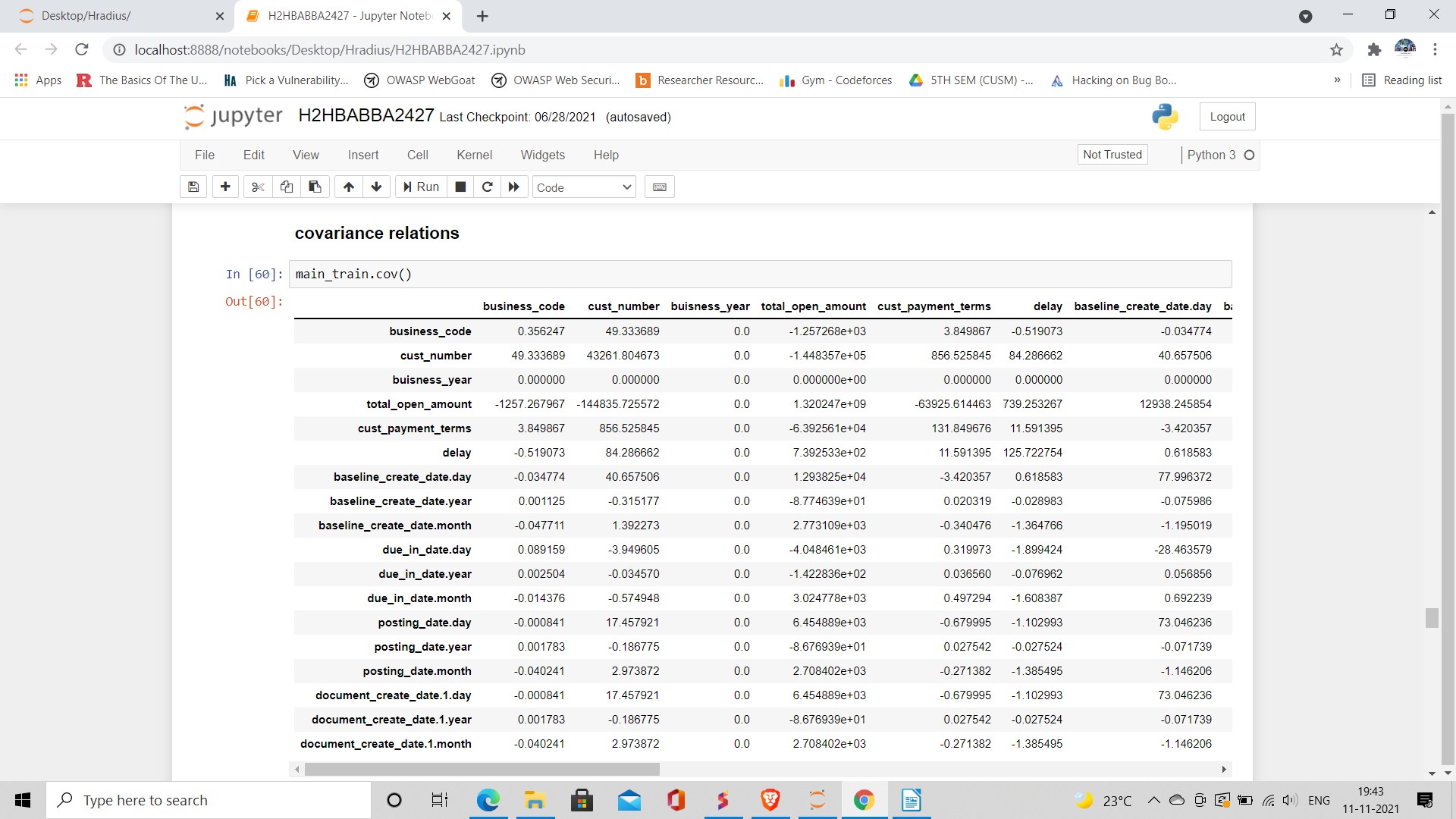
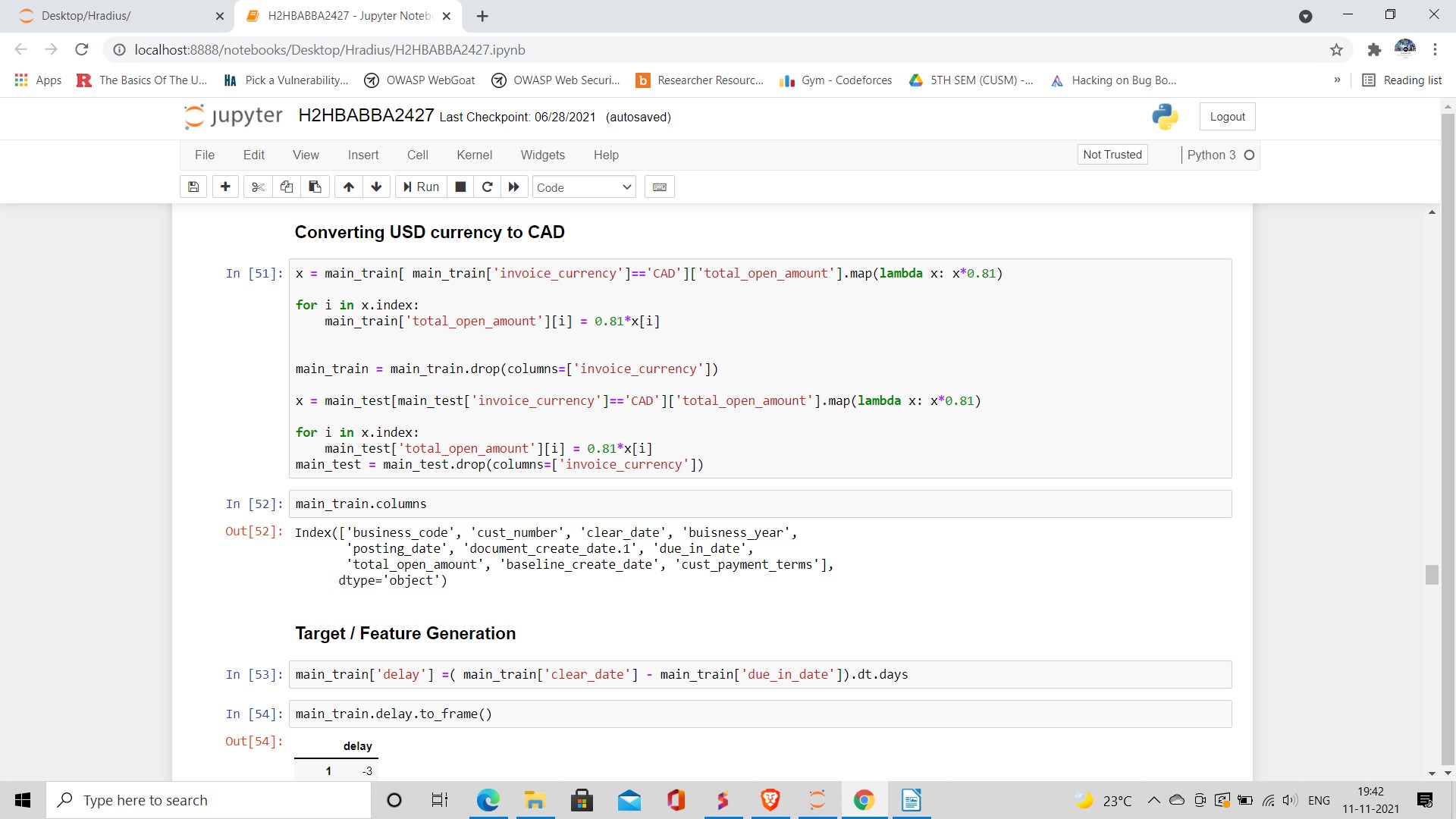
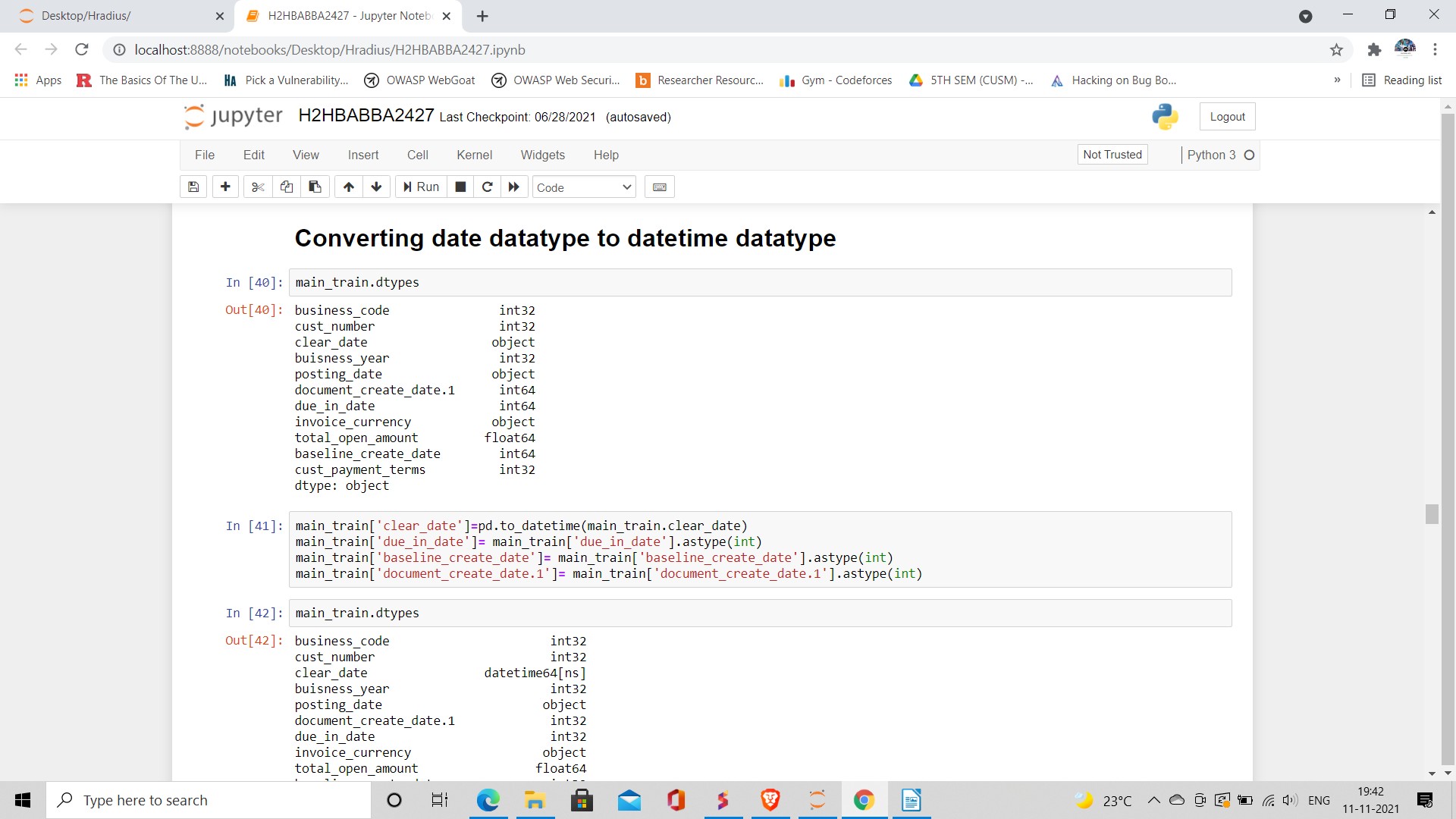
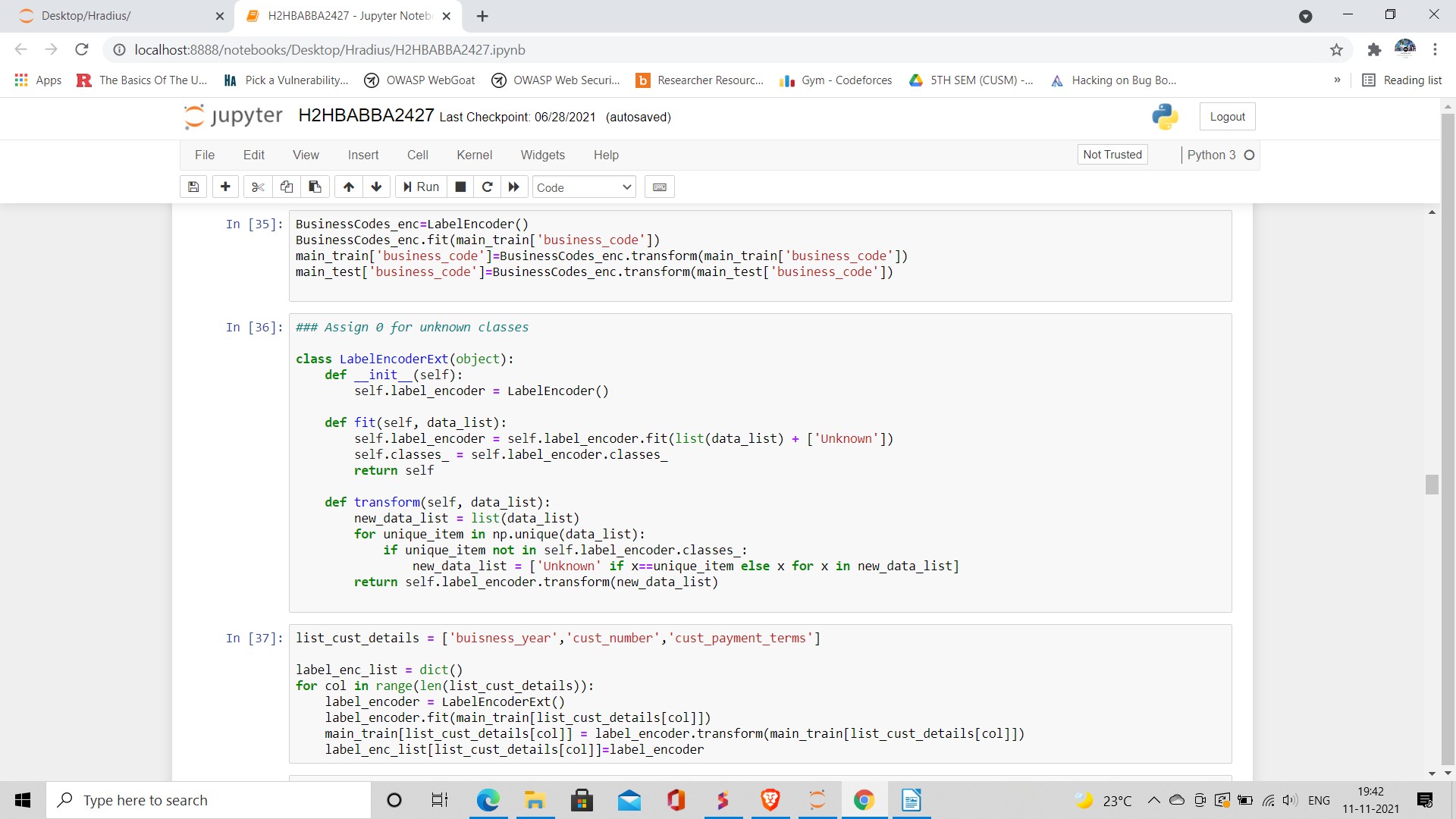
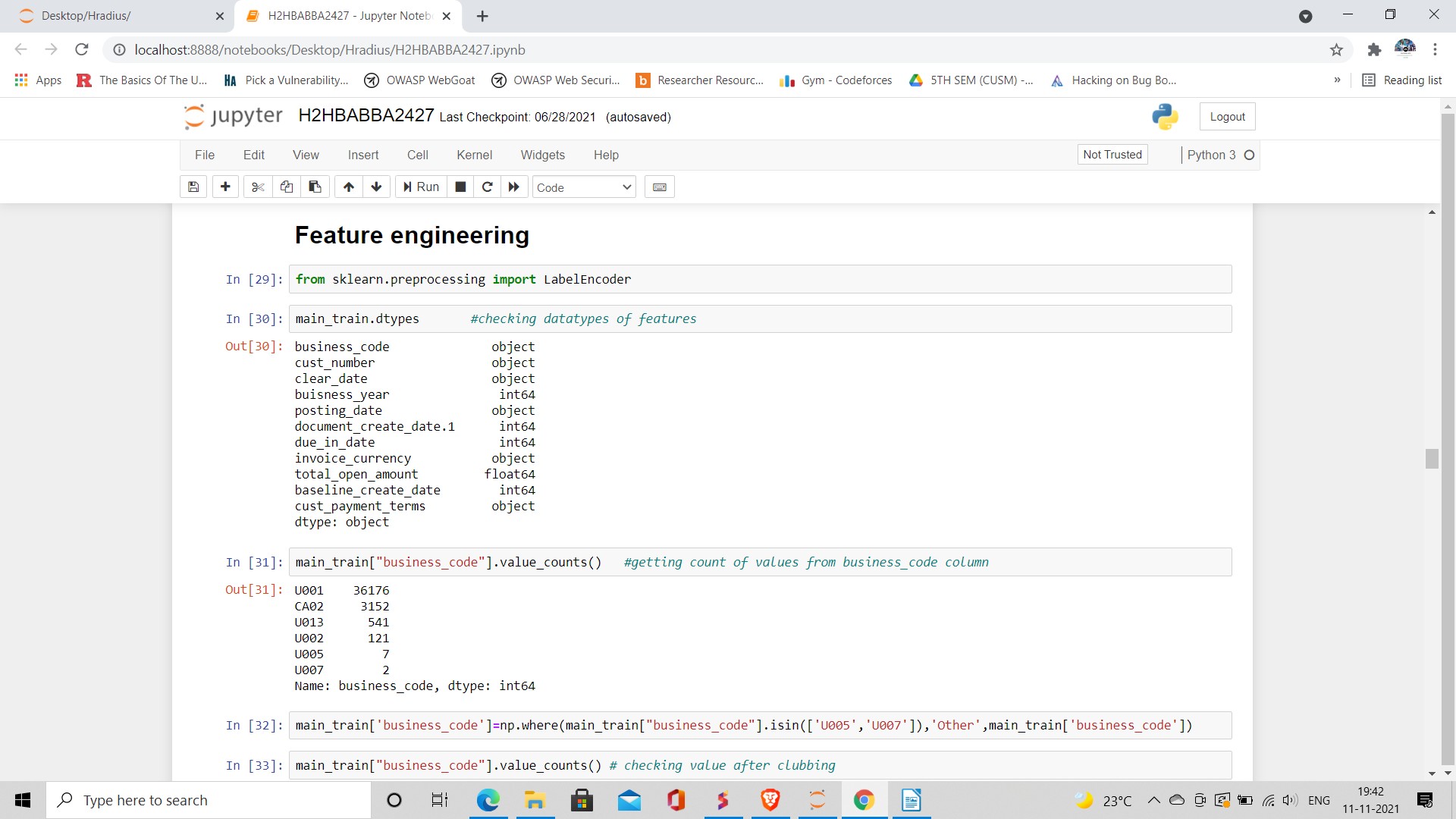
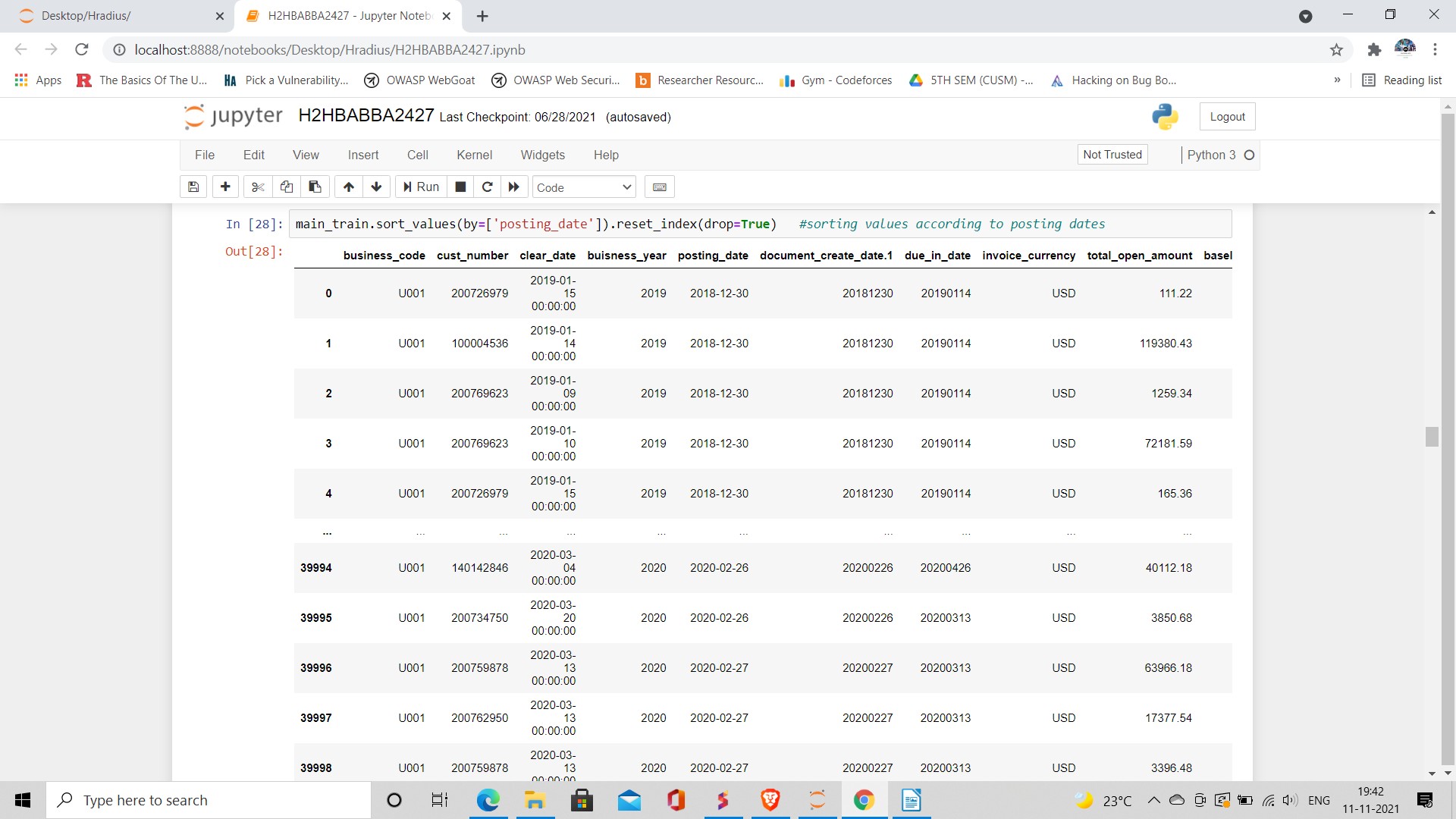
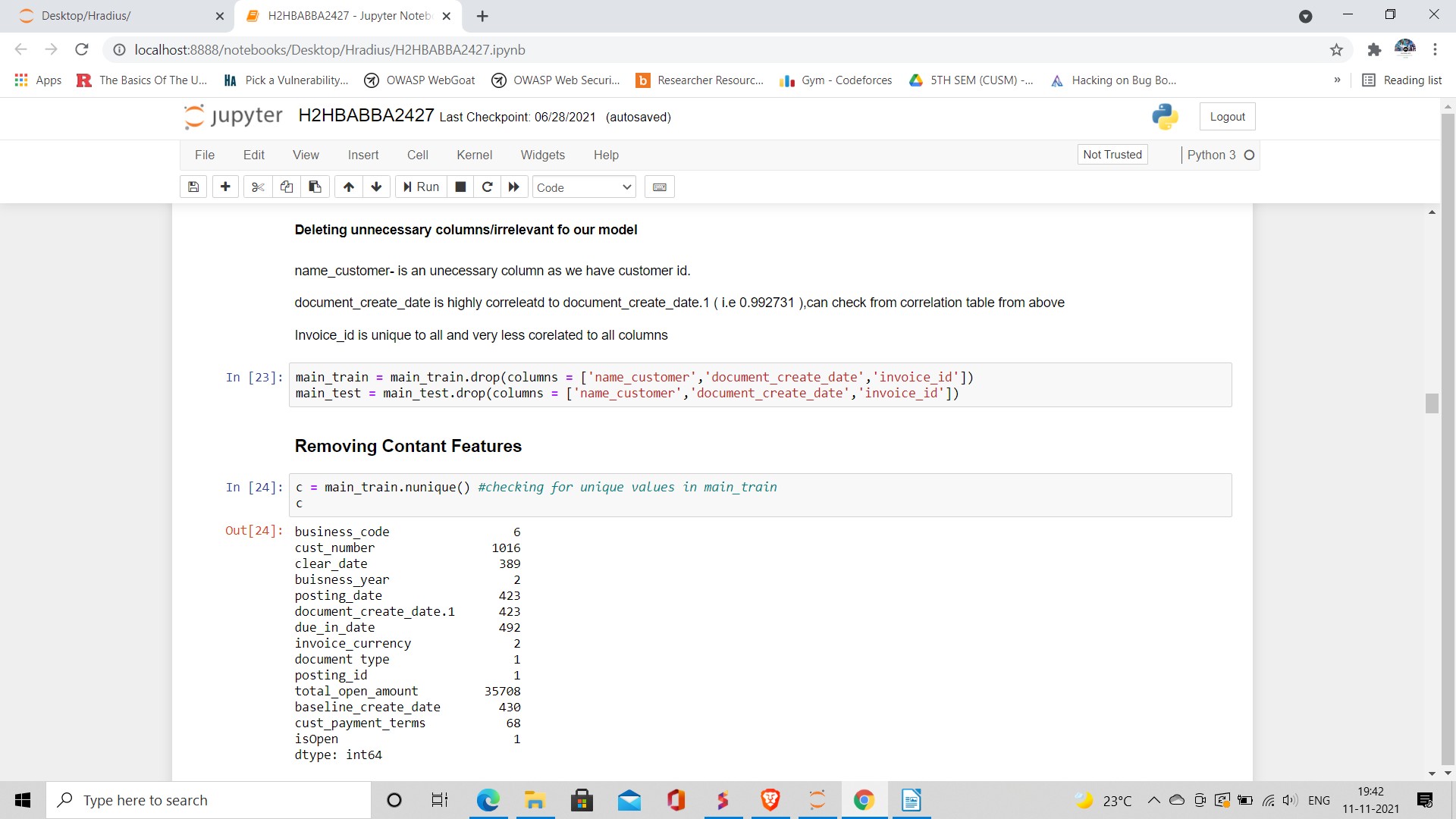
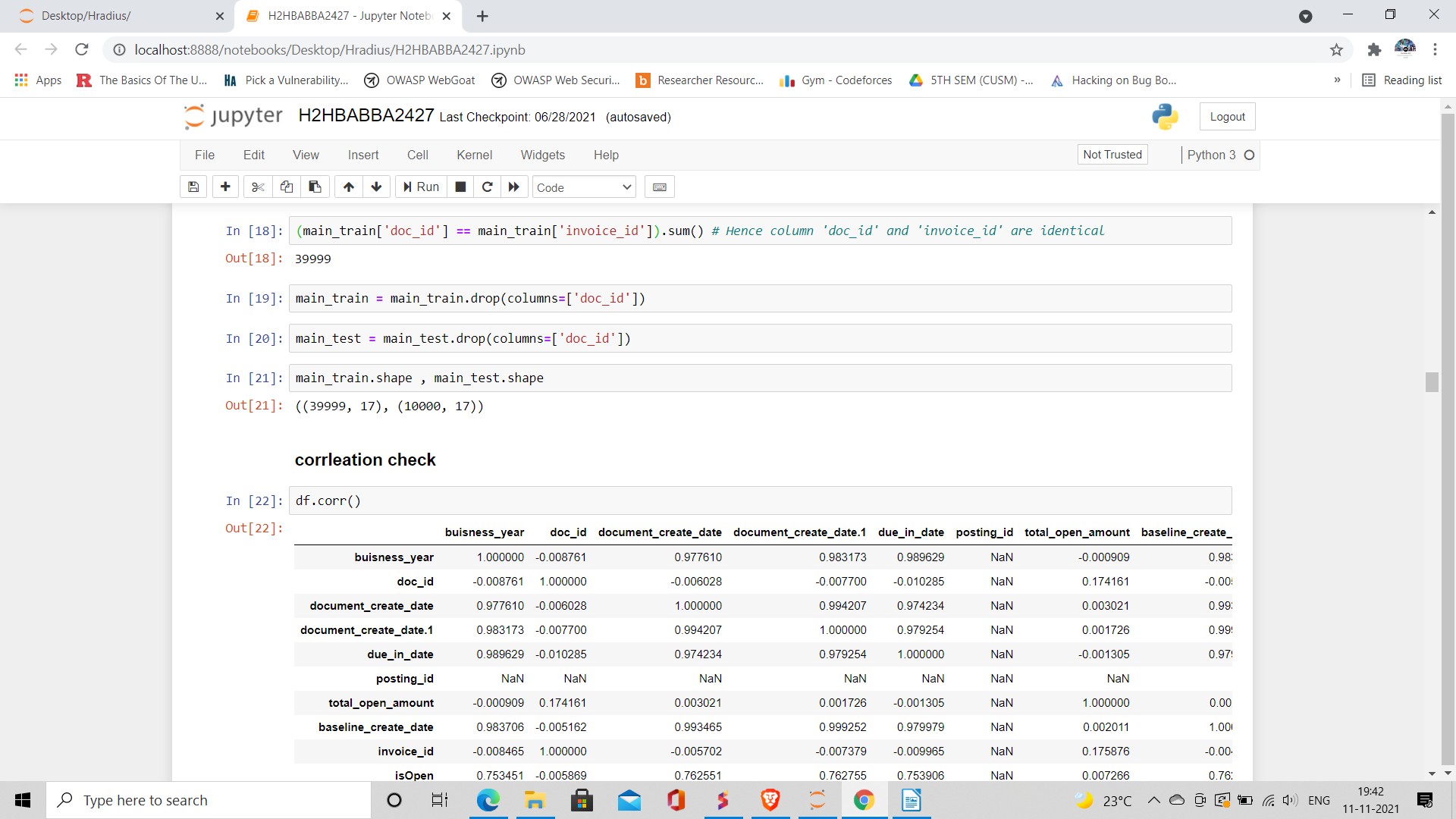
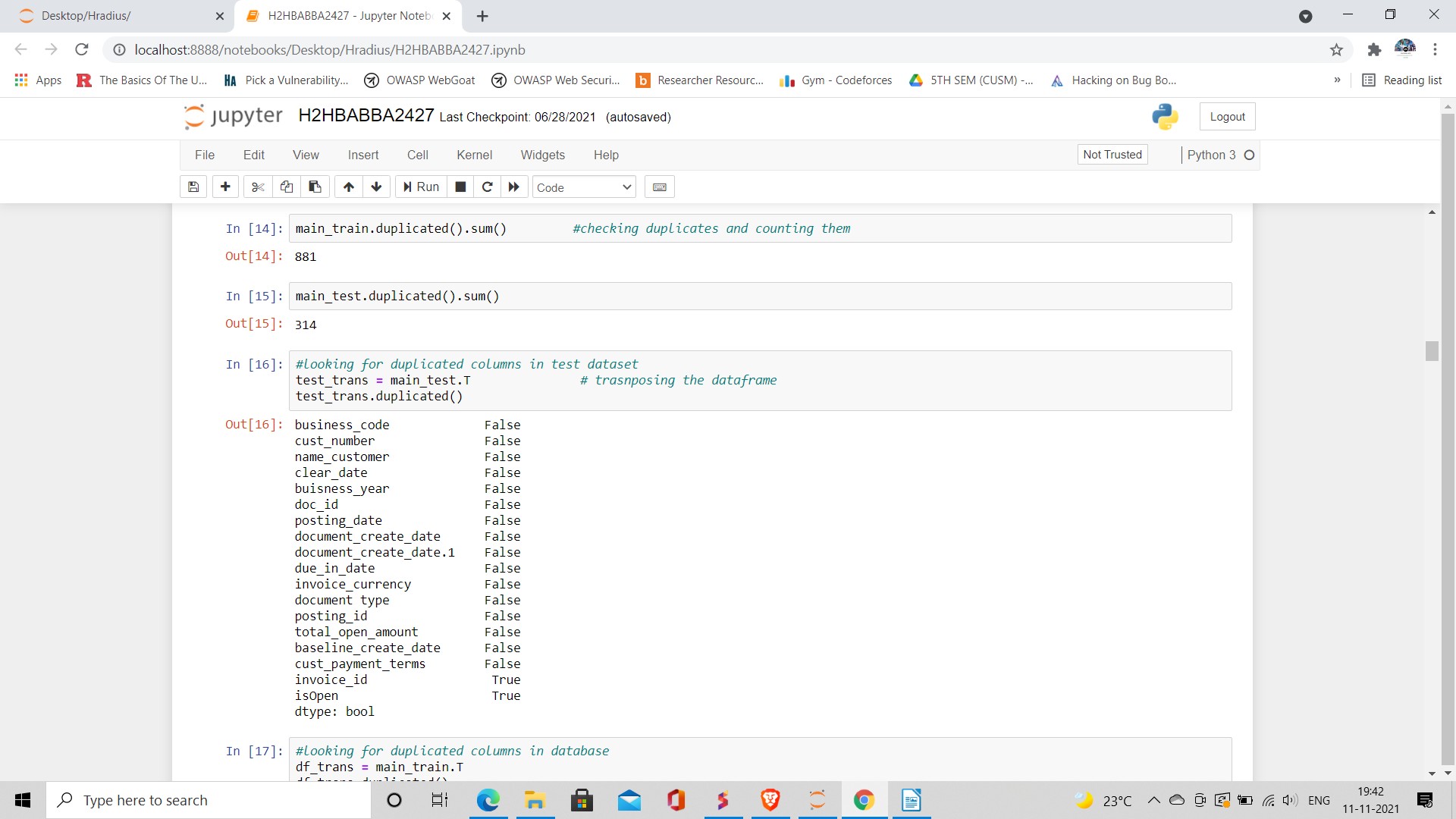
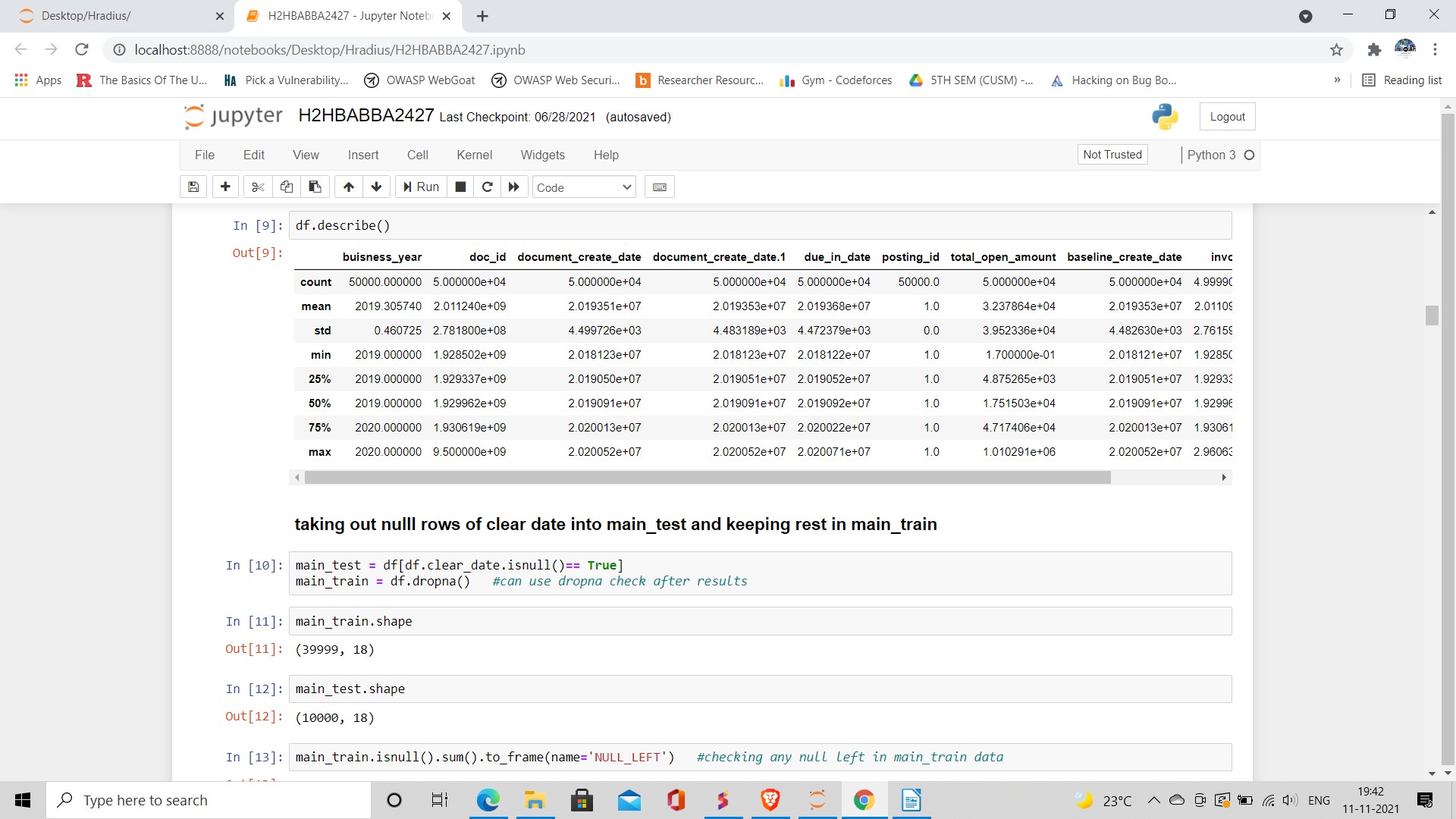
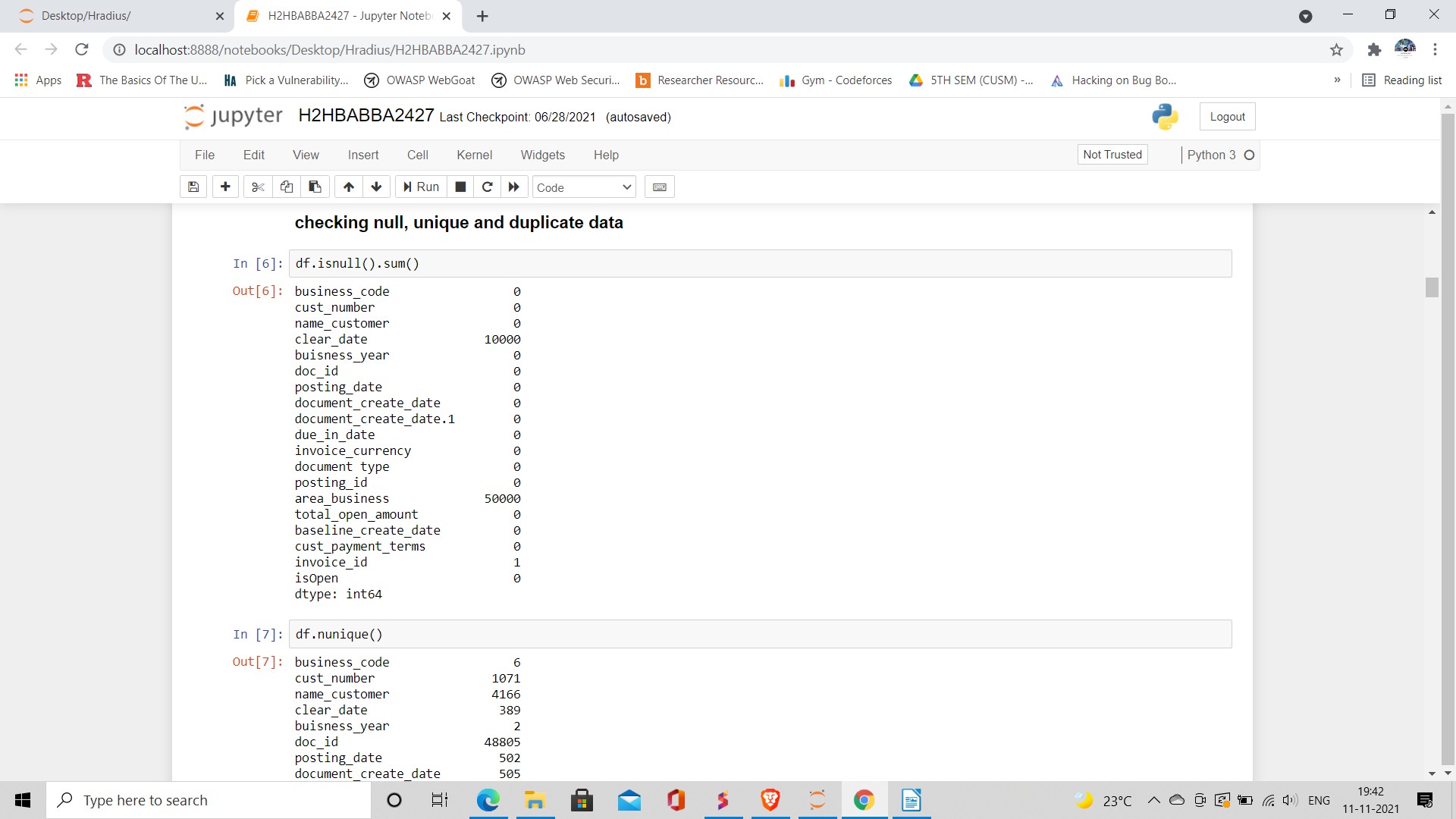
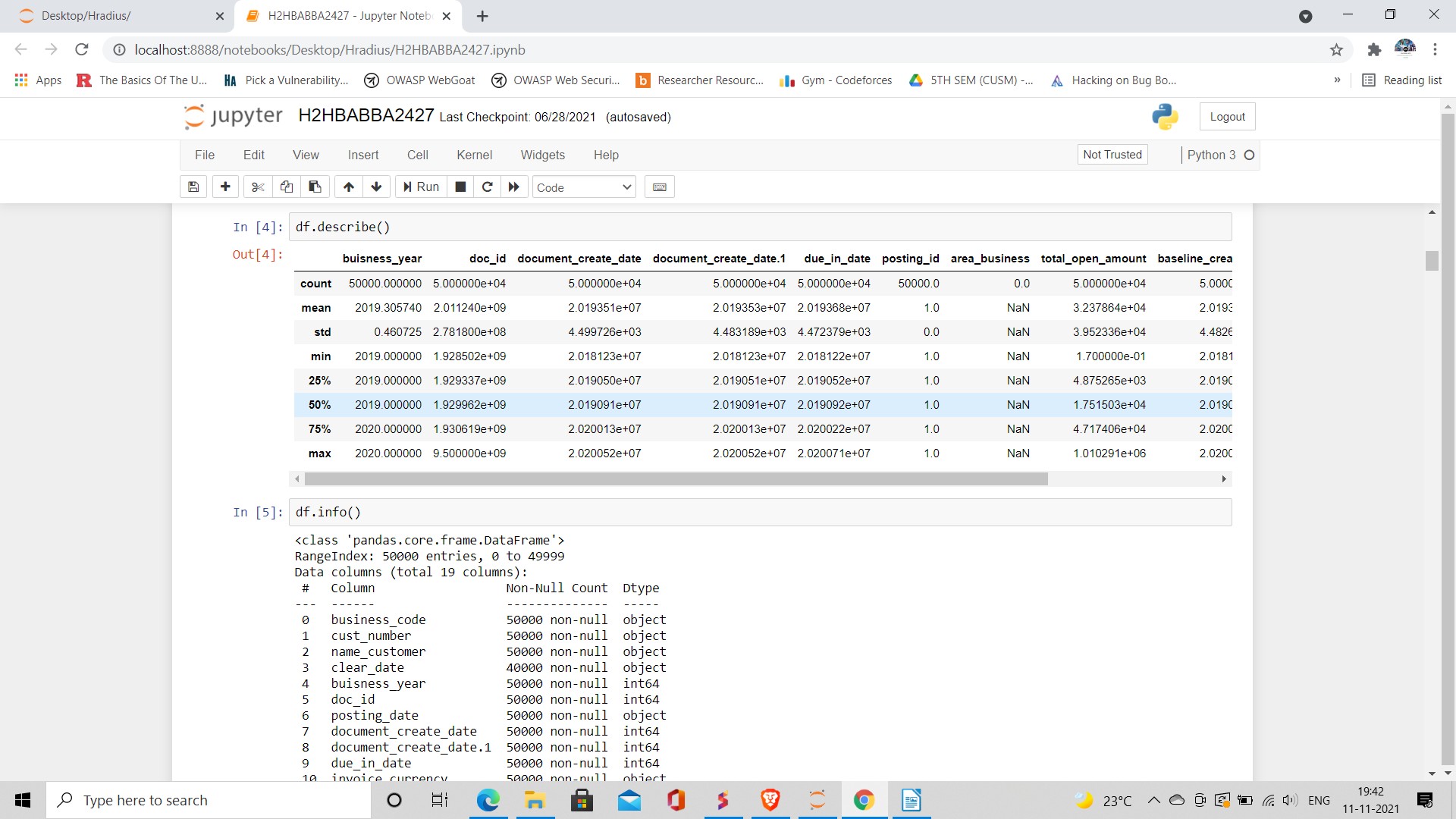
* + Building and Training our Model
  + Training and validating Data
  + Model Architecture
  + Building downloading system

# Innovations in Project:

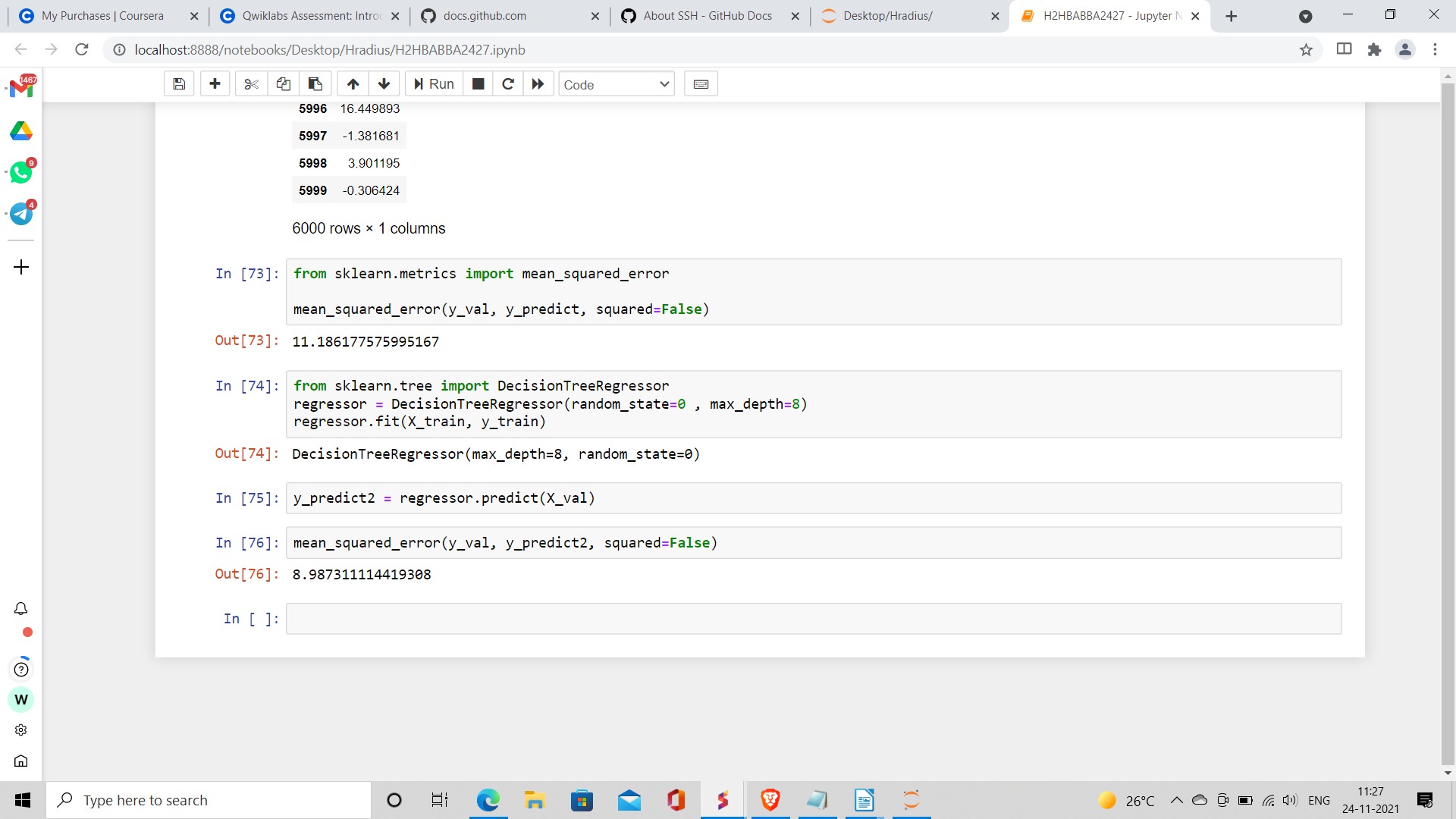
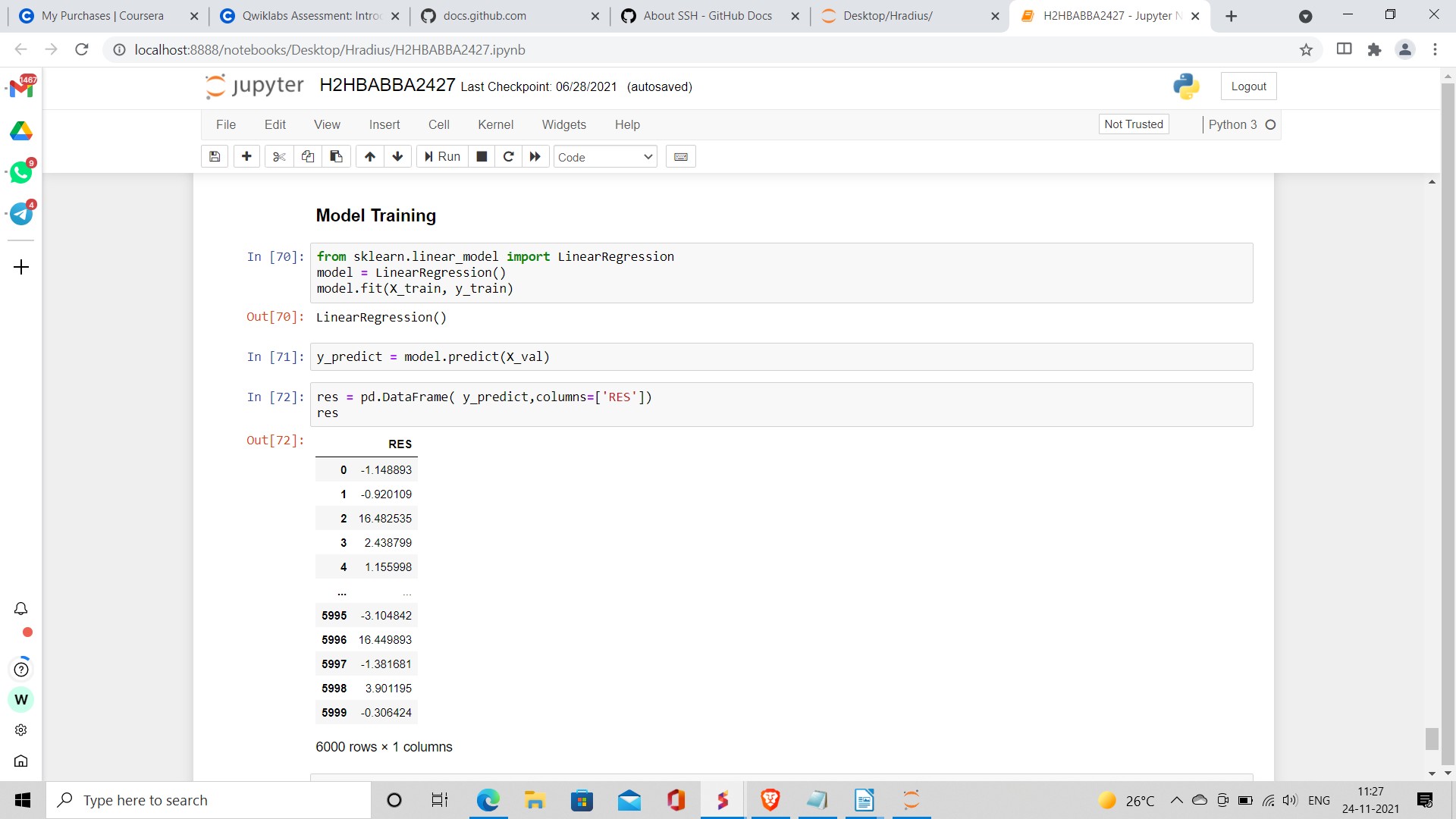
1. The given project can be turned into a full stack app or software where companies can input their data in CSV formats and a using a simple UI they would be able to predict the dates for the payment.
2. One more feature that we can innovate is the Auto reminder feature : As one inputs their data in the software and the predicted dates are months after so instead of checking everyday one can just process the data once and can create a reminder sheet which will be set off to call or email companies for the payment reminder.



# Implementation :



**Output :**



# Applications:

The described problem can essentially be split into two major parts:

1. Creating a network of Dutch SMEs from un-standardized and noisy data.
2. Improving the current methods of late payment prediction using features extracted from the network.

The thesis also provides an end-to-end solution, of how data about the business supply chains can be used to build a network of SMEs through entity resolution. Furthermore, it shows how this network can be leveraged through methods such as graph embedding, to improve the predictions of late-payments. The focus of this thesis is to see whether the addition of features extracted from a graph of related companies can improve the accuracy of late payment predictions. To do this we define the following research question and underlying sub-questions.

* MQ: Can graph features be used to improve the prediction of late invoice payments compared to currently popular methods? This is done by answering three separate sub-questions:
  + SQ1: How can a network of SMEs be built from data that is un-standardized, noisy and partial?
  + SQ2: How should the data and the graph be structured to be able to extract meaningful features?
  + SQ3: Does the addition of graph features improve the prediction of late payments?

# Tools Used :

## Jupyter Notebook

Is a project and community whose goal is to "develop open-source software, open- standards, and services for interactive computing across dozens of programming languages". It was spun off from IPython in 2014 by Fernando Pérez and Brian Granger. Project Jupyter's name is a reference to the three core programming languages supported by Jupyter, which are Julia, Python and R, and also a homage to Galileo's notebooks recording the discovery of the moons of Jupiter.

Jupyter Notebook can connect to many kernels to allow programming in different languages. A Jupyter kernel is a program responsible for handling various types of requests (code execution, code completions, inspection), and providing a reply.

Kernels talk to the other components of Jupyter using ZeroMQ, and thus can be on

the same or remote machines. Unlike many other Notebook-like interfaces, in Jupyter, kernels are not aware that they are attached to a specific document, and can be connected to many clients at once. Usually kernels allow execution of only a single language, but there are a couple of exceptions

## Tensorflow

TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.

TensorFlow was developed by the Google Brain team for internal Google use in research and production. The initial version was released under the Apache License 2.0 in 2015.

TensorFlow can be used in a wide variety of programming languages, most notably Python, as well as Javascript, C++, and Java.[11] This flexibility lends itself to a range of applications in many different sectors.

## Pandas

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

It is free software released under the three-clause BSD license.[2] The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals.[3] Its name is a play on the phrase "Python data analysis" itself.[4] Wes McKinney started building what would become pandas at AQR Capital while he was a researcher there from 2007 to 2010.

## Library features

* 1. Data Frame object for data manipulation with integrated indexing.
  2. Tools for reading and writing data between in-memory data structures and different file formats.
  3. Data alignment and integrated handling of missing data.
  4. Reshaping and pivoting of data sets.
  5. Label-based slicing, fancy indexing, and subsetting of large data sets.
  6. Data structure column insertion and deletion.
  7. Group by engine allowing split-apply-combine operations on data sets.
  8. Data set merging and joining.