WRITE-UP STORY
HEART DISEASE PREDICTION

As a part of our COMP 840 ML project, we started by searching for a suitable dataset on Kaggle website and applied filters to obtain the best dataset for our project which is the heart disease dataset of type classification. Initially, the dataset had 900 rows and 10 columns, but we performed feature selection on it and extracted the 7 best features from it. As AutoML training requires a minimum of 1000 rows for training, we added 100 more rows to avoid that conflict.

Next, we created a new project named COMP 840 ML project in the Google Cloud console and made sure that we have enabled billing for our project and enabled the Vertex AI APIs. We created a service account for week 11 lab and generated a JSON key file for Windows, which we utilized for our final project. In the Vertex AI section under the datasets page, we created a tabular dataset by specifying regression/classification objective and set the region as the US Central1. We then added the heart.csv file into it and started training an AutoML classification model using GCP tools. Initially, our training failed due to an insufficient number of rows in the dataset. Later on, we added some more rows to work fine in GCP, which took almost 2-3 hours to train our model. We deployed our model to an endpoint.

We utilized Jupyter notebook as a workspace to test our model endpoint externally with online prediction using some instances. We created a main.ipynb file and imported our heart dataset and JSON key file into our notebook. Then, we included a client python program to test the endpoint externally and tested it by giving the endpoint ID along with our project ID. We gave some instances in a list of dictionaries format, and our model worked successfully. Later, we performed some basic statistics on our dataset, such as head(), info(), column(), describe(), isnull(), valuecounts(), etc. Performing all those basic statistics gave us a better understanding of our dataset.

Afterward, we used the same dataset to train a customized model using sklearn classifiers discussed in class. We wrote a function that takes in a custom-trained model, project, and instances as input and predicts their output using the Google Cloud AI Platform Prediction Service. The predictions are returned as a dictionary. Our dataset has both categorical and numerical features, so we used encoding techniques to convert categorical to numerical features. Our model has a total of 7 features and one target column, which are 'Age', 'ChestPainType', 'RestingBP', 'Cholesterol', 'RestingECG', 'MaxHR', 'ExerciseAngina', and heartdisease, respectively. We then split our dataset into training (80%) and test (20%) subsets using sklearn's train_test_split() function along with this we did target balancing. We then created a pipeline with a simple imputer, min-max scaler, and decision tree as a classifier to perform multiple functions simultaneously. We then did preprocessing on numerical and categorical features and performed column transformation on it because we have both categorical and numerical features in our dataset. For numerical features, to fill all the null values, we used simple imputer and also used min-max scaler, and for categorical features, we used a simple imputer to fill the null values and one-hot encoder to encode the categorical features.

Then we fitted the pipeline to the training data to make predictions on both the training and test datasets and calculated and printed evaluation metrics, including the confusion matrix, classification report, and accuracy score. The evaluation results are printed separately for the training and test datasets. We imported the matplotlib library to plot the Receiver Operating Characteristic (ROC) curve and Area Under the Curve (AUC) score for the binary classification model represented by the pipeline.The roc_curve and auc functions from sklearn.metrics were used to calculate the necessary values for the plot.

We used GridSearchCV from the sklearn.model_selection library to perform a grid search on the parameters of a given pipeline using the specified parameter grid. The grid search was performed with 3-fold cross-validation, prints a summary of the results and uses all available processors.

After that, we fitted the pipeline again and obtained predictions on the test data. We plotted the ROC curve again, and we found that the AUC had decreased after hyperparameter tuning.Next, we generated a pickle file by importing the pickle library and then fitted the pipeline again. We evaluated the pipeline on the test data and made a prediction using a dataframe, which was successful.

Finally, we performed custom training by passing some instances in 2D. We also did custom ML training in GCP by loading the model.pkl file into a bucket and then deploying it to the created endpoint. After 30 minutes, our model was successfully deployed. We then predicted our model by giving sample instances in GCP under Deploy and Test. We also locally tested our model in Jupyter Notebook by giving the endpoint ID and project ID. It worked successfully both in GCP and Jupyter.

Throughout our project, we also performed some basic statistics on our dataset, such as head(), info(), column(), describe(), isnull(), and valuecounts(). Performing all these basic statistics gave us a better understanding of our dataset.

In summary, we obtained a heart disease dataset and performed feature selection to extract the best features from it. We then trained an AutoML classification model using GCP tools, deployed it to an endpoint, and tested it externally using Jupyter Notebook. We also trained a customized model using Sklearn classifiers, performed hyperparameter tuning, and evaluated the model using evaluation metrics. Finally, we performed custom training and deployed our model in GCP and tested it successfully both in GCP and Jupyter Notebook.

Things found helpful:

- The documentation regarding GCP which is provided in the modules really helped us while working in GCP.
- Sklearn site also found to be helpful while performing hyperparameter tuning in choose best parameters and its syntax.
- Concept of generating pickle file helped us alot which ultimately reduced our time for training the model always.
- Taking a dataset with less number of rows saved alot of time during training.

Things found unhelpful:

- We found performing hyperparameter tuning unhelpful because after performing tuning our ROC got decreased.

Summary of lessons learned:

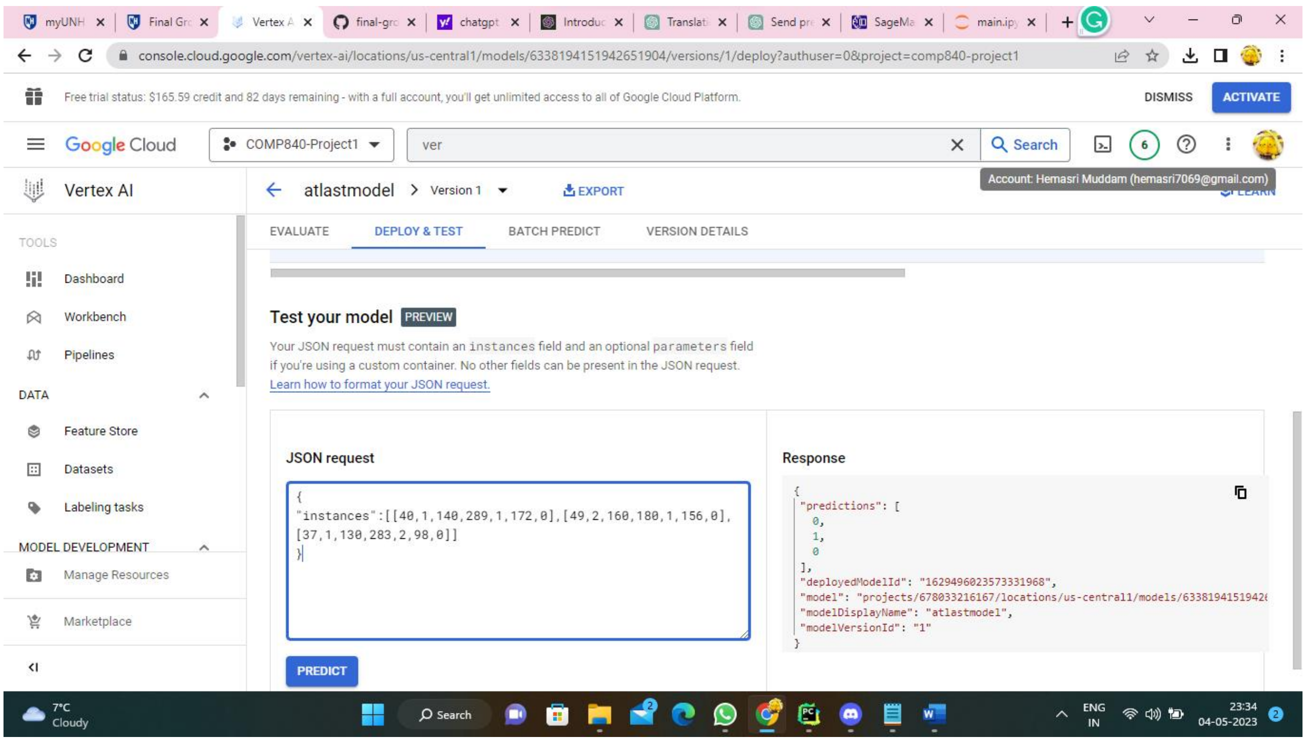
- Our clear and consistent communication helped to stay aligned on project goals, progress, and challenges.
- Establishing clear roles and responsibilities, as well as timelines and deadlines, helped us to ensure that both of us are on the same page.
- Encouraging open and constructive feedback among us helped to identify and address issues early on in the project.
- Celebrating small wins and successes along the way helped us to maintain motivation and momentum throughout the project.

Another chance:

If we get a chance to do it again we would have implemented a feature selection technique to select best features among all to improve the performance of our model without selecting them manually.

Screenshots:

Contributor 1 Hemasri Muddam:


console.cloud.google.com/vertex-ai/locations/us-central1/models/6338194151942651904/versions/1/deploy?authuser=0&project=comp840-project1

Free trial status: $165.59 credit and 82 days remaining - with a full account, you'll get unlimited access to all of Google Cloud Platform.

Account: Hemasri Muddam (hemasri7069@gmail.com)

Test your model PREVIEW

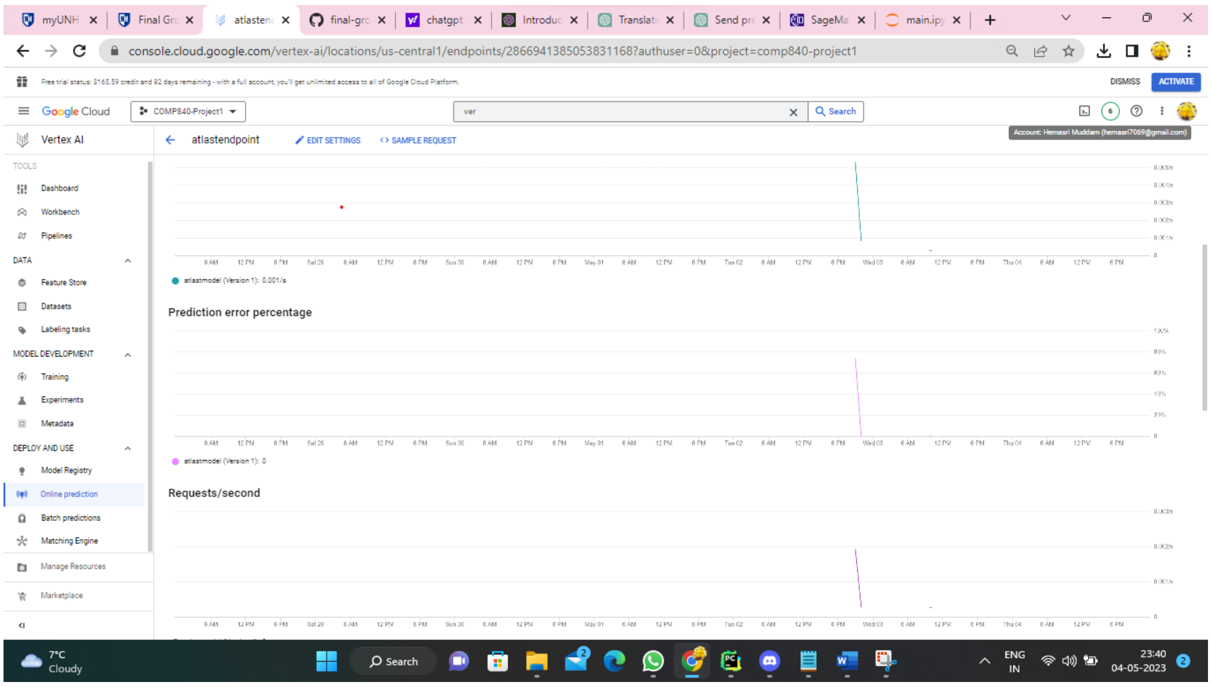
Your JSON request must contain an `instances` field and an optional `parameters` field if you're using a custom container. No other fields can be present in the JSON request.
Learn how to format your JSON request.

JSON request

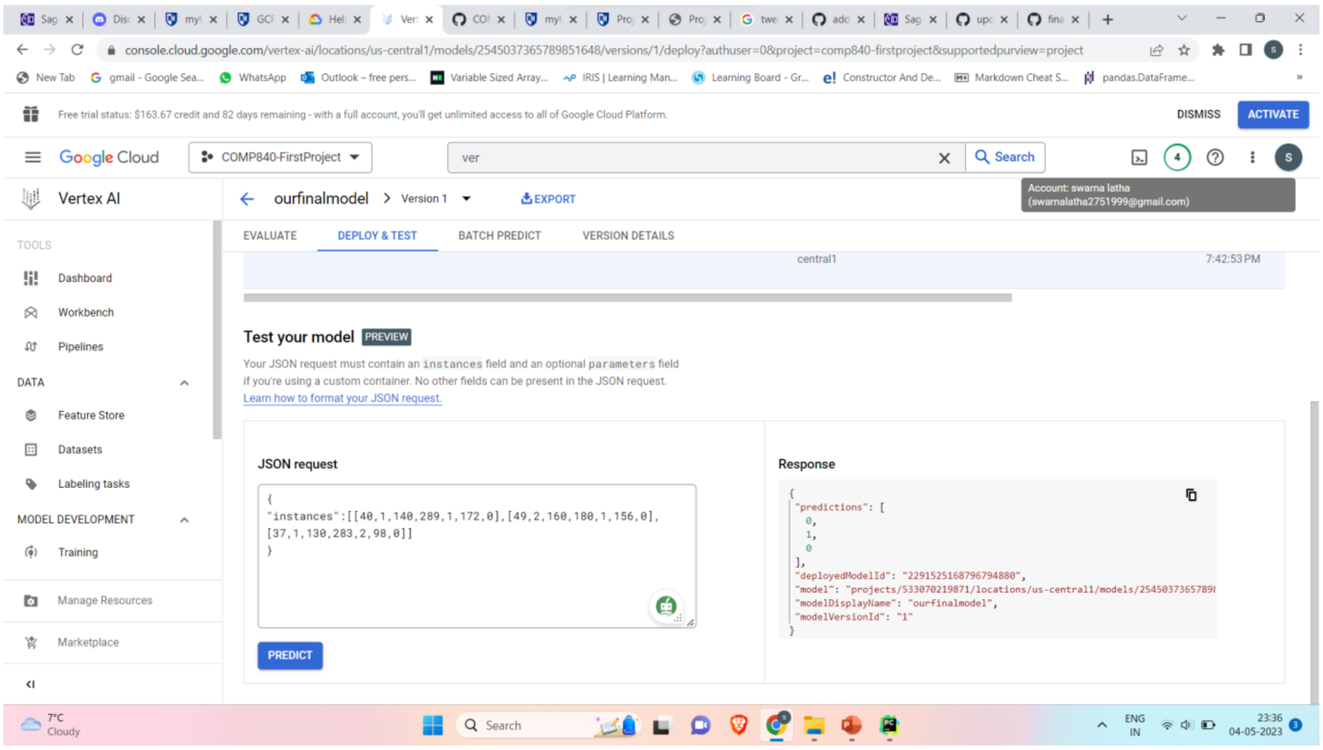
```
{
"instances":[[40,1,140,289,1,172,0],[49,2,160,180,1,156,0],
[37,1,130,283,2,98,0]]
}
```

Response

```
{
 "predictions": [
   0,
   1,
   0
 ],
 "deployedModelId": "1629496023573331968",
 "model": "projects/678033216167/locations/us-central1/models/6338194151942
 "modelDisplayName": "atlastmodel",
 "modelVersionId": "1"
}
```



Contributor 2 Swarnalatha Anugu:


console.cloud.google.com/vertex-ai/locations/us-central1/models/2545037365789851648/versions/1/deploy?authuser=0&project=comp840-firstproject&supportedpurview=project
Free trial status: $163.67 credit and 82 days remaining - with a full account, you'll get unlimited access to all of Google Cloud Platform.
DISMISS
ACTIVATE
Google Cloud
COMP840-FirstProject
Vertex AI
Account: swarna latha (swarnalatha2751999@gmail.com)
ourfinalmodel
Version 1
EXPORT
EVALUATE
DEPLOY & TEST
BATCH PREDICT
VERSION DETAILS
central1
7:42:53 PM

Test your model PREVIEW

Your JSON request must contain an `instances` field and an optional `parameters` field if you're using a custom container. No other fields can be present in the JSON request.
Learn how to format your JSON request.

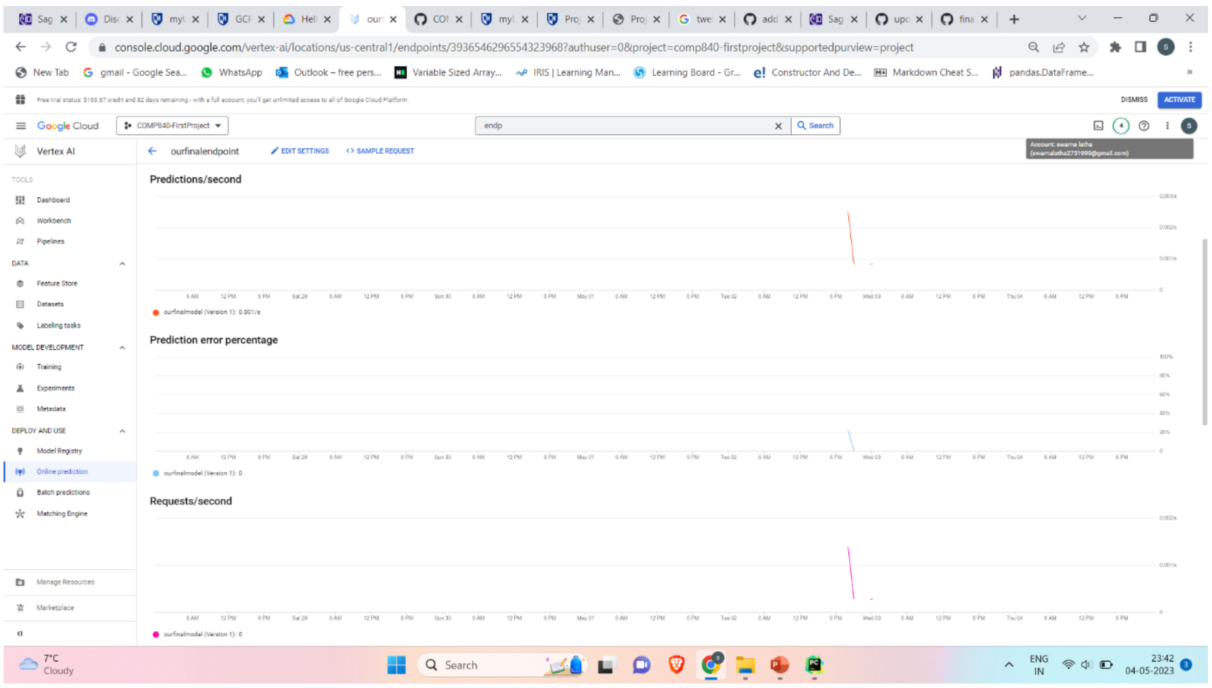
JSON request

```
{
"instances":[[40,1,140,289,1,172,0],[49,2,160,180,1,156,0],
[37,1,130,283,2,98,0]]
}
```

PREDICT

Response

```
{
 "predictions": [
   0,
   1,
   0
 ],
 "deployedModelId": "2291525168796794880",
 "model": "projects/533070219871/locations/us-central1/models/2545037365789
 "modelDisplayName": "ourfinalmodel",
 "modelVersionId": "1"
}
```



Authors,

Hemasri Muddam

Swarnalatha Anugu