SANTA CLARA UNIVERSITY
LEAVEY SCHOOL OF BUSINESS

CHOICE OF MLOPS - MLFLOW vs VERTEX AI

(Project Final Report)

For the fulfillment of the course
Software Project Management
As part of MSIS program at
Santa Clara University - Leavey School of Business

By

Ankitha Shetty
Kirtana Kirtivasan
Krishna Sai Tejaswini Kambhampati
Swetha Vijaya Raju(Team Leader)

TABLE OF CONTENTS

EXECUTIVE SUMMARY

COMPANY OVERVIEW:

Lead is a mid-size company founded in 2009 with a mission to provide a platform for communities, to not only provide real time navigation on roads, but also reports accidents, traffic jams, road works, community driven points of interests and many more activities that are the root cause of traffic congestion in cities. The company currently has 150 employees. Currently, Lead is operating only in the Bay Area, California with 2 types of users – 2000 active users who passively relay information about driving speed and traffic conditions whenever they have the application open in the background. Lead relies on large scale data solutions on a daily basis as they attempt to solve problems like - Predicting ETA, Matching Riders & Drivers and Serving the right advertisements. There are multiple engineers (Data scientists, Software engineers, reliability engineers, Data engineers and many more) working on opportunistically including ML solutions for their application.

PROBLEM SUMMARY:

Lead uses multiple ML models to explore and develop their ML algorithms for various use cases; performing semi-manual operations for training, validation and deployment; limited monitoring and validation capabilities after a model is deployed; and a hideously long deployment cycle from idea to production – This is a challenge that manifested itself causing a chaotic state at Lead. Currently, Lead does not have a service to manage the ML lifecycle.

PROJECT OBJECTIVES:

The major objective of this project is to streamline the process and minimize the timelines for deployment by reducing the manual work wherever possible in the life cycle of the algorithm, and comparatively reducing overall costs of the system using managed services. Invest in deployment processes through managed services (such as MLFlow or Vertex AI) and ensure that, by using these streamlined systems, a single Data Scientist could be able to close the product cycle from research to production.

RECOMMENDATION:

Vertex AI

● Vertex AI provides a unified platform that can accommodate data readiness to ML model deployment streamlining the entire ML lifecycle while MLFlow will only help in Experimentation Model tracking.

● With Vertex AI, a single Data Scientist could close the product cycle from research to production.

● Vertex AI natively uses Google infrastructure and our system is majorly on Google Cloud. So the adoption of Vertex AI to our current system is compatible.

SUCCESS CRITERIA:

● The adoption of Vertex AI will bring down the operational cost of ML models in Lead by 34% in the next 5 years.

● With the adoption of Vertex AI, deployment timeline will be reduced by 22% approximately.

LIFE CYCLE OF A ML ALGORITHM AT LEAD[2]

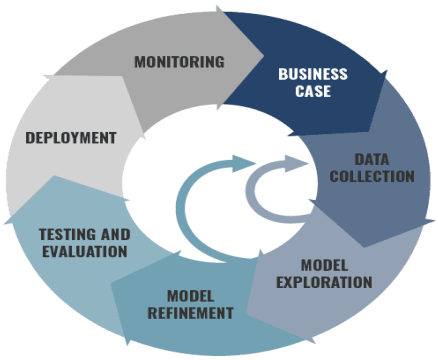


Figure 1: Life Cycle of a Machine Learning Algorithm(Image Credit)

All Machine Learning (ML) projects in general are highly iterative in nature; They might also need to circle back to earlier phases after completing a phase. Moreover, a project isn't complete after shipping the first version; we might have to get feedback from real-world interactions and redefine the goals for the next iteration of deployment.

Figure 1 shows the life cycle of a ML algorithm with 7 framework activities and fits them into a development model called CRoss Industry Standard Process for Data Mining (CRISP-DM), to organize projects and also use it as a reference checklist while planning and implementing them. In this process model, each phase can be iterated several times, until satisfactory results are obtained. It is also possible to jump back to the previous phase when need comes.

The seven framework activities are – Defining the task, collecting data, model exploration, model refinement, testing and evaluation, deployment and integration, monitor and maintain.

WORKFLOW OF THE ALGORITHM FROM IDEATION TO DEPLOYMENT AT LEAD

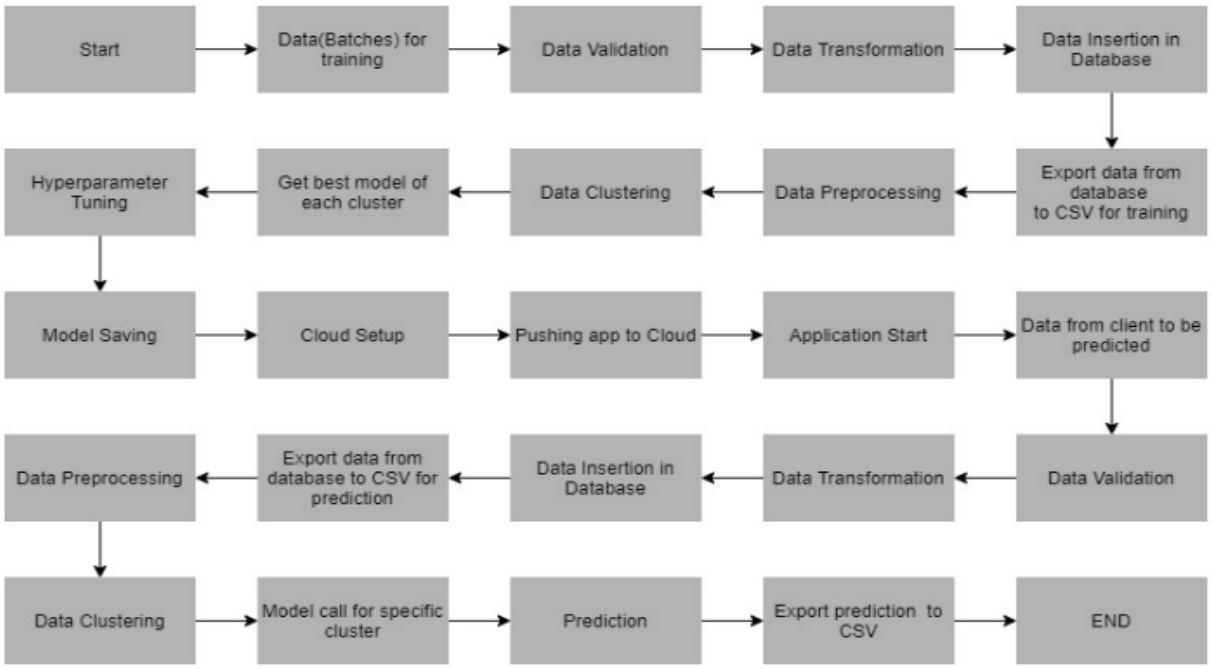


Figure 2 : Workflow of the model (Image-credit)

CHECKLIST FOR MANAGING ML PROJECTS AT LEAD:

We need to understand how lead currently works to understand the problem statement better**.**

Lead uses the following steps to strategically plan and implement machine learning projects successfully.

Step 1 – Planning and Project Setup - There are 5 important objectives of this step (4 -6 days)

1. Defining task - **Real time ETA prediction**
2. Determining project feasibility by analyzing the below factors (10 hrs)
    a. Cost of data acquisition for the task specified - **$0.1/GB of data to store in BigQuery**
    b. Cost of wrong predictions for the task - **loss of $20 per occurrence(approx)**
    c. Computational resources available both for training and inference
        i. Will the model be deployed in a resource-constrained environment - **Yes**
        ii. What are the latency requirements for the model - **Prediction latency under 10s**
3. Specifying project requirements (20 hrs)
    a. Establishing optimization metric
4. Identifying model tradeoffs(5 hrs)
    a. First model - Aim for a neutral first launch, focus spent on building the proper machine learning pipeline required for prediction

    b. Existing model - **Revisiting to meet updated information after discovering a promising general approach**
5. Setting project codebase (10 -15 hrs)
    a. Organizing codebase to modularize data processing, model definition, model training, and experiment management - **Uses Python , Docker and Kubernetes engine**

Step 2 – Data collection and labeling – Includes 4 primary objectives (~25 hrs - 2 days)

1. Define ground truth - This task enables labeling data and ensuring ground truth is well defined as quality of the data labels has a large effect on the upper bound of a model's performance- **past eight weeks average speed of a segment on a particular time (2 hrs)**
2. Building data ingestion pipeline - **From Kafka and Google Bigquery**
3. Validating data quality (15-20 hrs)
4. Revisit step 1 and ensuring data is sufficient for the task (5 hrs)

Step 3 – Model exploration – The primary objective of model exploration is to (~5-7 days, longer for new projects)

1. Establish baselines for model performance -
    a. First a simple model is built using initial data pipeline
    b. This simple model is overfit to training data
    c. Other parallel (isolated) ideas are explored at this stage
    d. Find State of the Art model for the problem domain (if available) and reproduce results, then apply to dataset as a second baseline
    e. Revisit Step 1 and ensure feasibility
    f. Revisit Step a and ensure data quality is sufficient

Step 4 – Model refinement - As the team attains a general idea of successful model architectures and approaches for our use case; this step focuses on performance gains from the model with below major objectives (10 - 15 days)

1. Applying the bias variance decomposition for irreducible errors, avoidable bias, variance,and validation set overfitting (10 hrs)
2. Perform model specific optimizations (ie. hyperparameter tuning) and iteratively debug model as complexity is added (30-40 hrs)
3. Performing error analysis to uncover common failure modes - **Use clustering to uncover failure modes and improve error analysis** (25 -30 hrs)
    a. Select all incorrect predictions. (Optionally, sorting observations by calculated loss to find the most egregious errors)
    b. Run a clustering algorithm such as DBSCAN across selected observations
    c. Manually explore the clusters to look for common attributes which make prediction difficult
4. Revisit Step 2 for targeted data collection and labeling of observed failure modes

Step 5 – Testing and evaluation (5 -7days)

1. Different components of a ML product to test include – (20 - 25 hrs)
    a. Testing the full training pipeline (from raw data to trained model) which is running nightly, to ensure that changes haven't been made upstream with respect to how data from our application is stored
    b. Running inference on the validation data and ensuring model score does not degrade with new model/weights for every code push
    c. A quick functionality test that runs quickly (<5 minutes) ensures no broken functionality during development
    d. Setting alerts for downtime, errors and distribution shift in data (2 hrs)

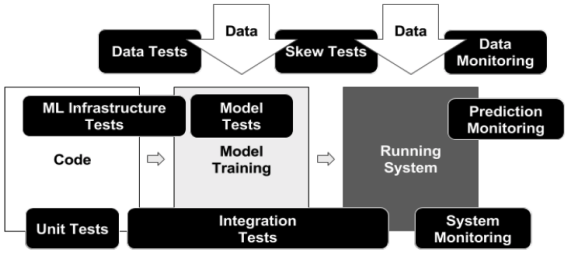


Figure 3 : ML based System Testing and Monitoring (Image credit)

2. Evaluation of multiple factors for Production readiness with current System Architecture (25 hrs)
    a. Data – Feature expectations are captured in a schema, ensuring cost of features is reasonable, Data pipeline has appropriate privacy controls, ability to add new features, all input feature code is tested
    b. Model – Model specs are reviewed and submitted, ensure offline and online metrics correlate, all hyperparameters have been tuned, calculated impact of model staleness
    c. Infrastructure – Ensure training reproducibility**,** Unit testing of model specifications, integration testing of pipeline, ensuring possibility of debugging model (if needed) before serving, model canariying[2] before serving, rolling back (if needed)
    d. Monitoring – **(Currently Manual) -** Monitoring dependency changes, numerical stability, regression of computing performance and prediction quality (~20 hrs)
    e. Revisit model evaluation metric; ensure that this metric drives desirable downstream user behavior

---

(superscript) 2 - a canary release is a synonym of A/B testing where a different product version is shown to a small percentage of users

Step -6 – Model deployment[3] - Focus of the current project at Lead

1. Package the system into a Docker container and expose a REST API for inference-**Manually setting up dependencies for packaging (5 hrs)**
2. Serving the new model to small subset of users (i.e. 5%) to ensure everything goes smoothly, then roll out to all users – **By performing A/B Testing** (Measuring the delta between the new and current model's predictions will give an indication for how drastically things will change when you switch to the new model)
3. Maintain the ability to roll back model to previous versions
4. Monitoring live data (functional and operational monitoring) and model prediction distributions - **Currently involves lot of human intervention and time (24*7 for first 15 days, once a day during peak hours for performance metrics)**

Step -7 – Ongoing model maintenance (continuous monitoring unless model is replaced)

1. Maintenance principles at Lead follows
    a. CACE (Changing Anything Changes Everything) Principle: As ML Systems are tightly coupled, changes to any feature-space, hyper parameters, learning rate etc. can affect model performance. So lead relies heavily on model validation tests that run every time code is pushed.
2. Periodically retrain model to prevent model staleness **(happens on a weekly basis)**
3. If there is a transfer in model ownership, educate the new team

Goal: The main focus of streamlining all the steps that can be automated along the cycle by adjusting the current system architecture without compromising the principles and objectives. All such steps are identified along the process and market research is done to find alternatives.

Deployment time: With the current system architecture the Data science team working on ETA prediction needs **~ 48-50 days** to deploy the updated ML model full stream to production.

TEAMS & ROLES DESCRIPTION

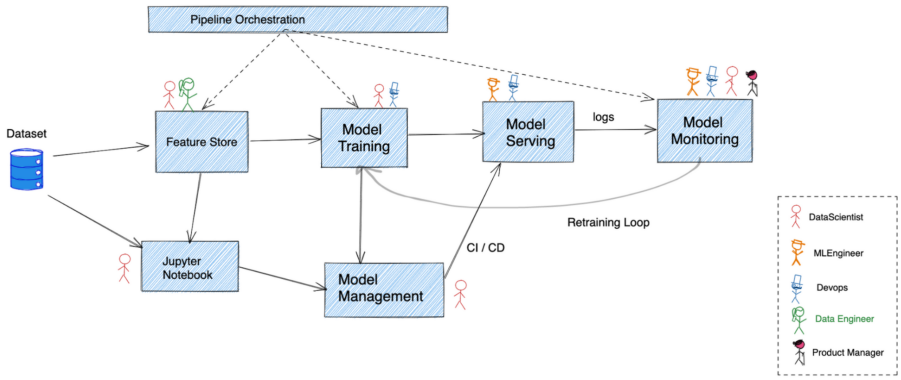


Figure 4 - Teams and roles in the overall pipeline (Image Credit)

A typical Data science team at Lead comprises of the following roles –

Stakeholders	User Story
Data Engineer	ETL / Build data pipelines to load, clean and process the data
Data Scientist	Data Refinement, Data Visualization, Data Analytics
ML Engineer	Build, train and deploy the Machine Learning model
Software Engineer	Integrating the ML model with the rest of the application
Product Manager	Making product decisions and point of contact with the client

Current Data Science team working on the use case includes - 1 Data Engineer, 3 Data Scientists, 2 Software Engineers, 1 ML Engineer and a Product Manager

CURRENT SYSTEM ARCHITECTURE

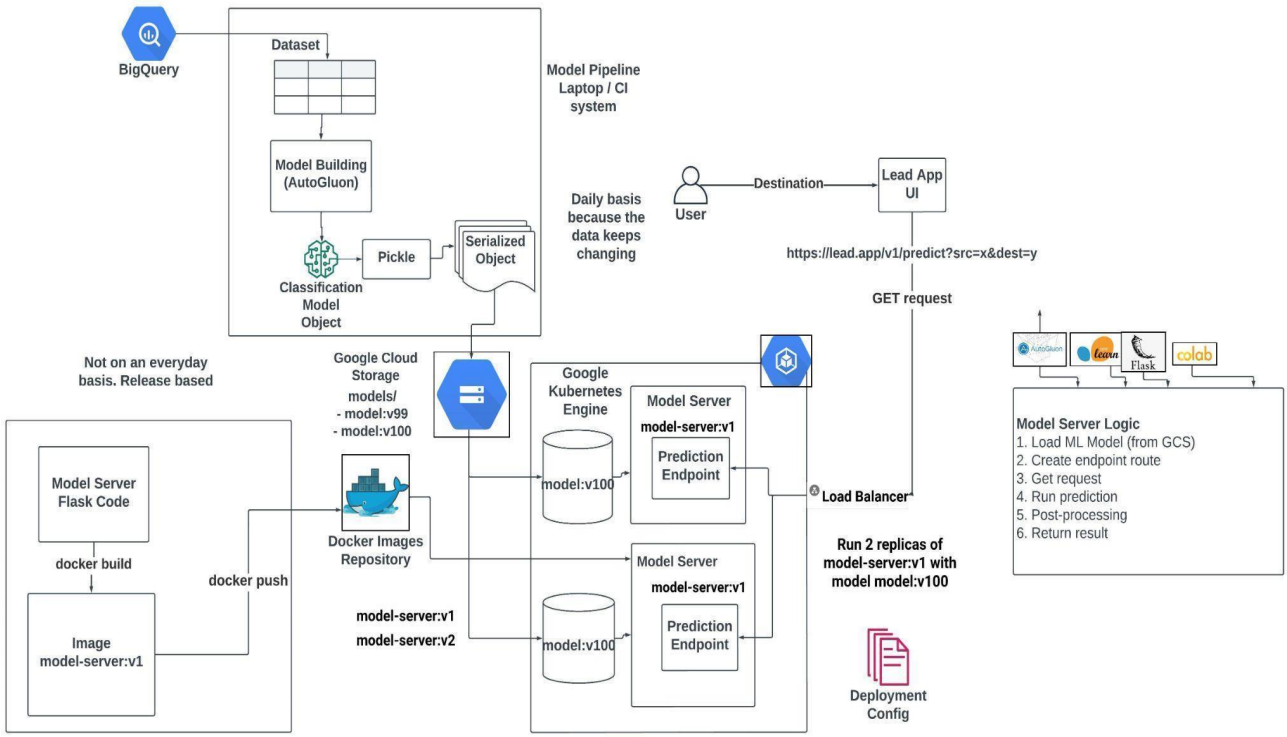


Figure 5 - Lead's current system architecture

Model Building:

Data Scientists load the datasets from BigQuery and build the model using preferred libraries (e.g. AutoGluon). The built model is serialized, versioned and stored in a GCS bucket.

Model Server:

Software Engineers parallelly build a web server with an endpoint to receive prediction results as HTTP requests. The model server is capable of loading the model stored in the GCS bucket and responds to the user results by asking the model to predict the result. The model server is built as a docker image and pushed to a docker registry (Docker Hub).

Deployment:

DevOps creates a Google Kubernetes Engine cluster with 3 nodes where Pods (containers) can be run. Once both the artifacts (model and model server) are built and pushed to their respective locations, the SREs (or Release Engineer) create a Deployment to run the model server of a specific version(e.g v1) with arguments to load a model of a specific version (e.g v100). If a new version of model server or model is to be deployed, the SREs (or Release Engineer) create a new Deployment with the new versions.

Working:

Once the model server deployment is complete, users via the Leads App can send requests to the endpoint with source and destination to get the prediction result.

Current Infrastructure

1. The instances used for serving the prediction results to users have the following configuration of at least (per instance),
    a. CPU: 8 cores
    b. Memory: 32GB
    c. Disk: 2TB (enough to hold the model + overhead)
    d. Network: 1Gbps upload/download rate
2. A load balancer to spread the load among different machines
3. The model server responds to http requests for prediction results

CURRENT OPERATIONAL COSTS[4]

Current Architecture

Training[5]

- 8 hours per model
- Per hour pricing
    - n1-standard-8, 8 CPU, 32 GB Memory = $0.44 / hr
    - NVIDIA_TESLA_K80 GPU = $0.617500 / hr
    - ~$1.1 per hour = ~$8.8 per day per model
    - 8.8*365 = $3200 per year per model

Serving[5]

Per hour pricing

- n2-standard-8 = $0.45 / hr
- 1 machine $0.45 / hr = ~ $4,000 per year (if running all the time)
- Scale up to 10 machines during a hypothetical two-hour peak that occurs everyday
    - 9 machines $0.45 per hour = ~ $3,000 per year (2 hrs per day)
- Total = $7,000

Engineering / deployment

- Initial CI/CD pipeline setup cost = $25,000

Costs for deploying 1 model to production

Model Infrastructure	Kubernetes clusters with a load balancer	$7,000
Data Support	Independent data pipeline manager for continuous update of analytic data	$10,000 (labor) + $3,200
Engineering / Deployment	Continuous integration and continuous deployment (CI/CD) system to pull model from registry	$24,500 (labor) + $516
Total Investment / yr		$34,500(labor) + $11,000

5 year TCO	$93,500

Costs for deploying an additional model to production

Model Infrastructure	Cluster can be adjusted to run multiple models according to changing using demands	+ $3,781 / yr
Data Support	Modules can be added to pipeline management system	+ $2,250 (labor)
Engineering / Deployment	CI/CD system can be extended to pick up additional model	+ $2,900 (labor)
Total Investment / yr		$5,150(labor) + $3,781 / yr
5 year TCO		+ $24,055

1. Cost for deploying 1 model to production = $93,500 over first 5 years
2. Cost for deploying 2 models to production = $118,000 over first 5 years
3. Cost for deploying 100 models to production = $2.5 million over first 5 years

Goal: To comparatively reduce the overall cost of Deployment by the choice of managed service

BUSINESS REQUIREMENTS

1. The Lead application should be available 24/7
2. The prediction error percentage [3] should be less than 0.1%
3. The prediction model shouldn't return outdated information and should be retrained in case
4. Accuracy rate depends on the requirement but usually an accuracy 85% and precision above 95% is acceptable
5. The adopted managed service should be compatible with the existing system

FUNCTIONAL REQUIREMENTS

1. The deployment of a new trained model needs to be under 30 minutes
2. The model latency should be less than 5 seconds
3. The overhead latency should be less than 5 seconds (Preferably deployed to west-coast servers)
4. The prediction model should be available 24/7 with 0 downtime with at least 3 instances of the model
5. Must be able to roll back to the older models

MARKET RESEARCH

Vertex AI[9]

It is a managed machine learning platform offered by Google.It provides you with all of Google's cloud services in one place to deploy and maintain AI models. It helps ML workflow by providing services like: Creating a dataset and upload data, Train an ML model on your data, Upload and store your model in Vertex AI or Deploy your trained model to an endpoint for serving predictions.

Key Features:

1. Dataset: The data type can be an image, tabular, text, or video and the source can be CSV from local drive, Cloud Storage or Big query. This is very useful to import your data, connect to different datasets with GCP.
2. Feature Store: If you have created some complicated features for a dataset and are looking to build a different type of code using the same data or want to re-use the features then you can add it in the feature store and add it rather than redoing the processes again (**reduction in 20 -25 hrs of manual labor**)
3. Labeling tasks :request for a manual labeling of the images by providing the labels with correct instructions(google provides sets of guidelines for creating good instructions) in a pdf file. End user receives an email when the task is completed. You can review the progress of the data labeling task in the Google Cloud console from the Labeling tasks page. This is a paid service
4. Training: Once data for the model is uploaded you can start the training. You can train models on Vertex AI by using AutoML, or if you need the wider range of customization options available in AI Platform Training, use custom training.AutoML: Create ML models without using any code (**reduction in 5 -7 days of manual labor**)
5. Vertex AI experiments: This helps to track and analyze different run experiments. Also enables to group and compare multiple models across experiment runs

6. Metadata: Analyze ML experiments to compare the effectiveness of different sets of hyperparameters.
7. Model registry: Contains list of all the successfully trained model
8. Endpoint**:** components expose the functionalities of the Vertex AI.
9. Model deployment for prediction: You can deploy models on Vertex AI and get an endpoint to serve predictions on Vertex AI whether the model was trained on Vertex AI.
10. Vertex AI pipelines: Pipeline components are self-contained sets of code that perform one part of a pipeline's workflow, such as data preprocessing, data transformation, and training a model which help to to streamline the development and execution of ML processes
11. Vertex AI Vizier: Used to tune hyper parameters in ML models (**3 -5 days in model refinement**)
12. Vertex AI prediction: A service optimized to run your data through hosted models
13. Use Vertex Explainable AI to get comprehensive model feature attributions and evaluation metrics.

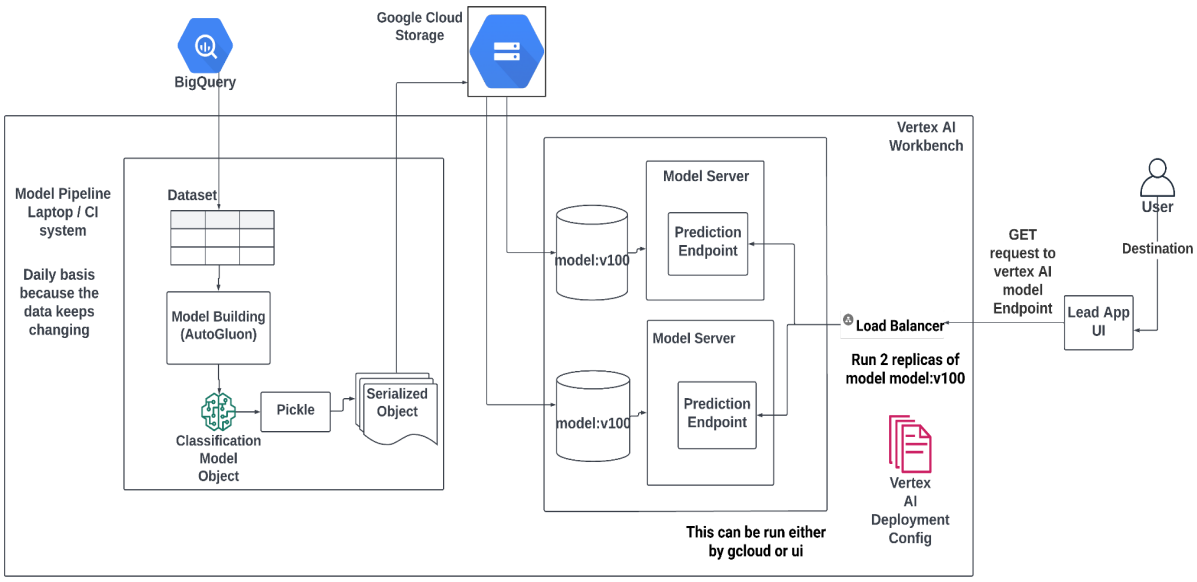


Figure 6 - Vertex AI integration with the existing architecture

Advantages:

1. Easy to access, takes as low as 30 seconds to login and access Vertex AI
2. New customers get $300 in free credits to spend on Vertex AI.
3. Vertex AI provides the user with a dashboard which has consolidated details of the project
4. Data can be uploaded to the cloud quickly. Eg. 1 million records were uploaded in less than a minute. The uploaded data is stored on google cloud bucket.
5. Provides AutoML models that will predict/classify the data provided without any major cleanup. If the requirements are simple classification and team size is small and they do not specialize in ML then AutoML is the best choice. AutoML can also be used to create an initial proof of concept (prototype) or baseline model.
6. Very intuitive environment, training the data was very simple, and the user did not have to write any code. Users can simply select the column to predict and put the data.
7. Vertex AI monitors the training and notifies the users via email when there is a failure or success of model training. The message contains the error message for further analysis.
8. After training the data, Vertex AI provides the performance details of the trained data using confusion matrix and the losses.
9. The code can be integrated with Github.
10. User customized code (.ipynb) can be easily added to Vertex AI workbench by simple drag and drop.In custom training, you can select from among many different machine types to power your training jobs
11. Vertex AI provides ML monitoring capable of tracking metrics, boosting the observability of your project
12. Creating an endpoint is very simple. Steps are very clear and an endpoint is created in a couple of minutes. This can be used for batch and online prediction options.
13. Google cloud provides you with a cost management tool. This will help the administrator (project manager) to assign a budget for the project. It also provides a dashboard to summarize the spending.There are features to alert the users through emails.

14. In Depth training videos and codes on how to set up and run ML models on vertex this would be helpful since we would have to spend on external training. But the team's time will have to be allocated for the said training.

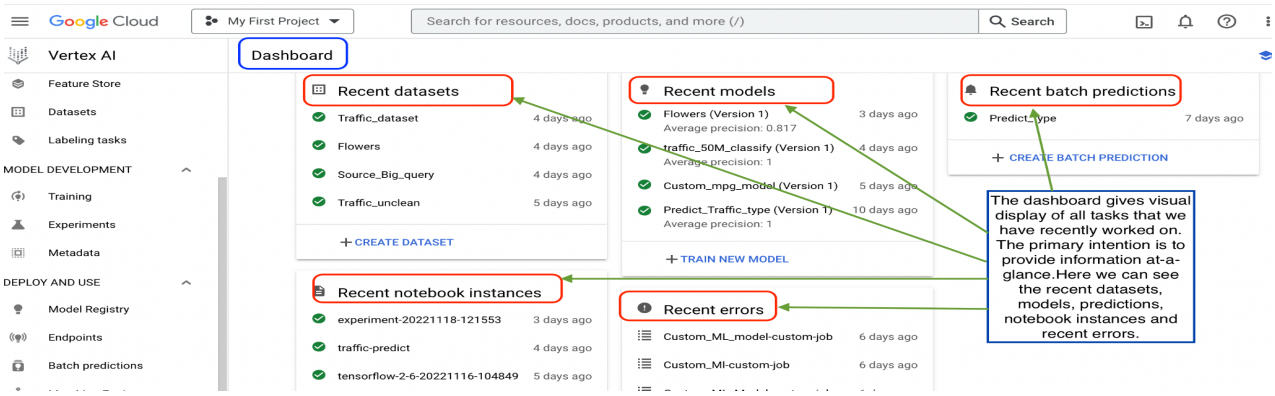
Disadvantages:

1. Vertex AI can become very costly if AutoML is used for model training. For tabular data[5],

Operation	Price per node hour (classification/regression)
Training	$ 21.252
Deployment & online prediction	Same as custom model ($0.39)

2. Vertex AI natively uses infrastructure from Google. The team members should be trained to use Vertex AI from data readiness to deployment.
3. The data scientists who will be using Vertex AI should have some familiarity about the cloud technologies like containerising, cloud buckets, etc.

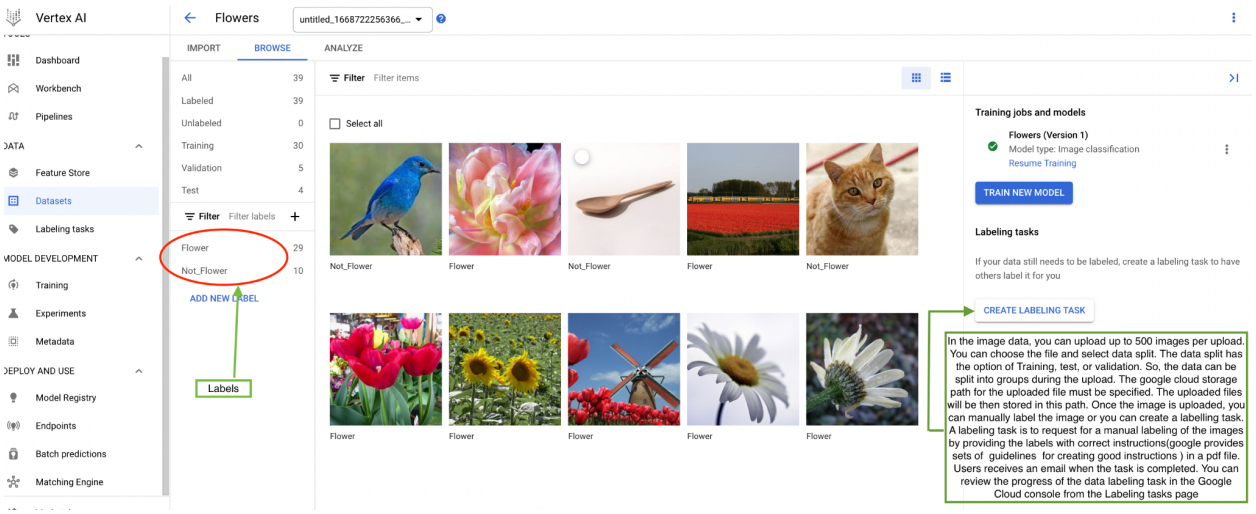
VERTEX AI Demonstration



[1] Vertex AI DashBoard

A. Data

[2] Vertex AI Datasets



[3] Vertex AI Labeling Task

B. Model Development

Vertex AI training tab

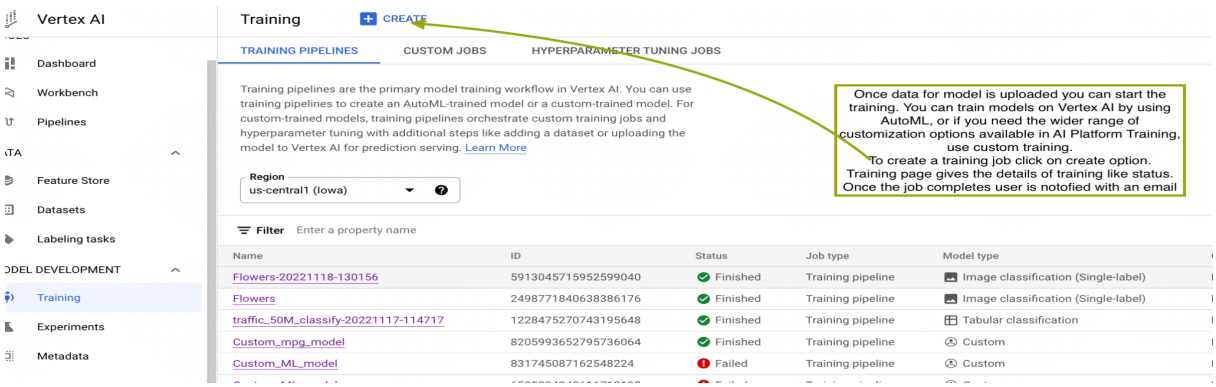
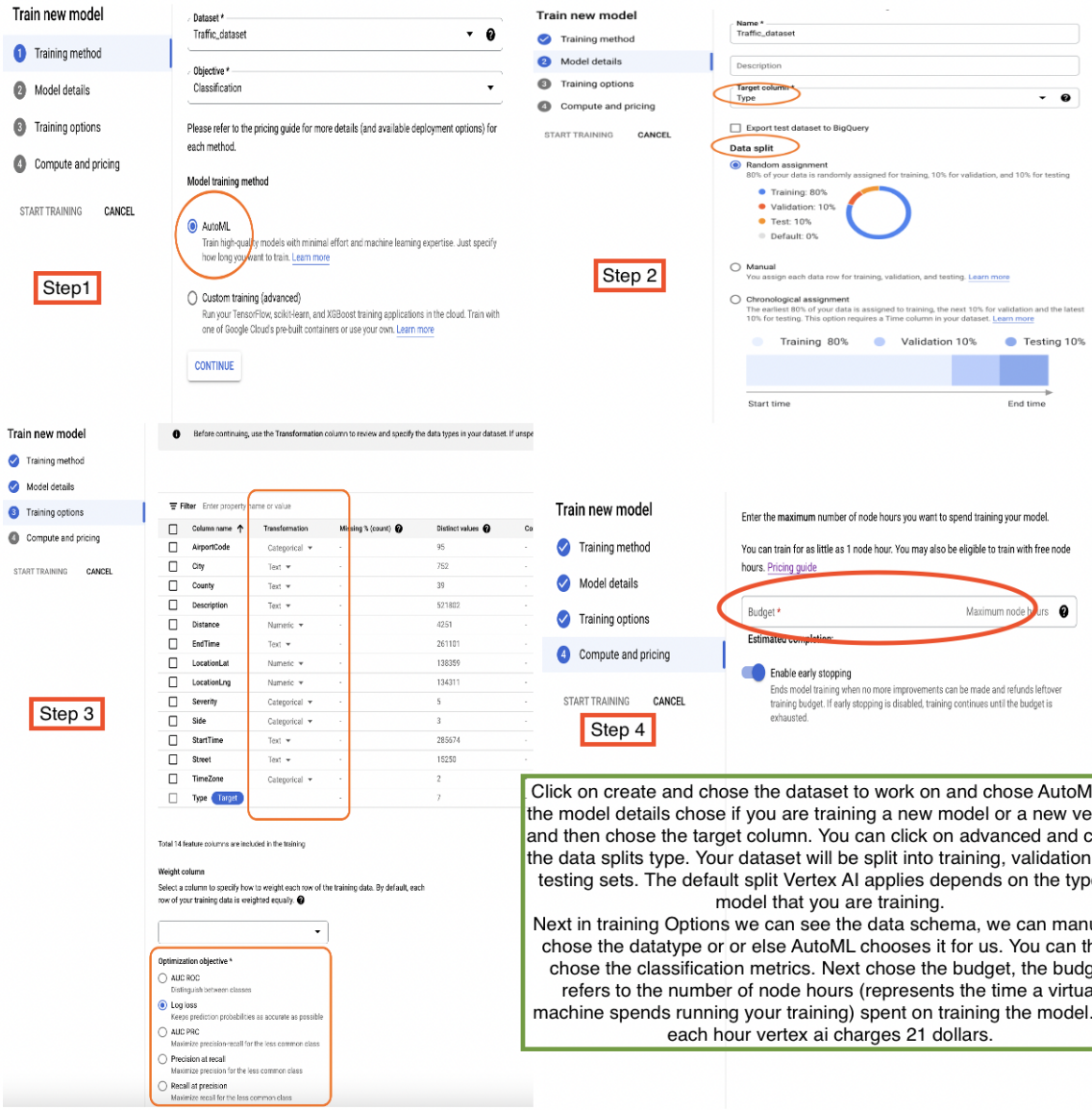


Image:[4]

Steps to create an AutoML model in Vertex AI[5]



Click on create and chose the dataset to work on and chose AutoML. In the model details chose if you are training a new model or a new version and then chose the target column. You can click on advanced and chose the data splits type. Your dataset will be split into training, validation, and testing sets. The default split Vertex AI applies depends on the type of model that you are training.
Next in training Options we can see the data schema, we can manually chose the datatype or or else AutoML chooses it for us. You can then chose the classification metrics. Next chose the budget, the budget refers to the number of node hours (represents the time a virtual machine spends running your training) spent on training the model. For each hour vertex ai charges 21 dollars.

Image:[5]

Custom Training:

Workbench[6]

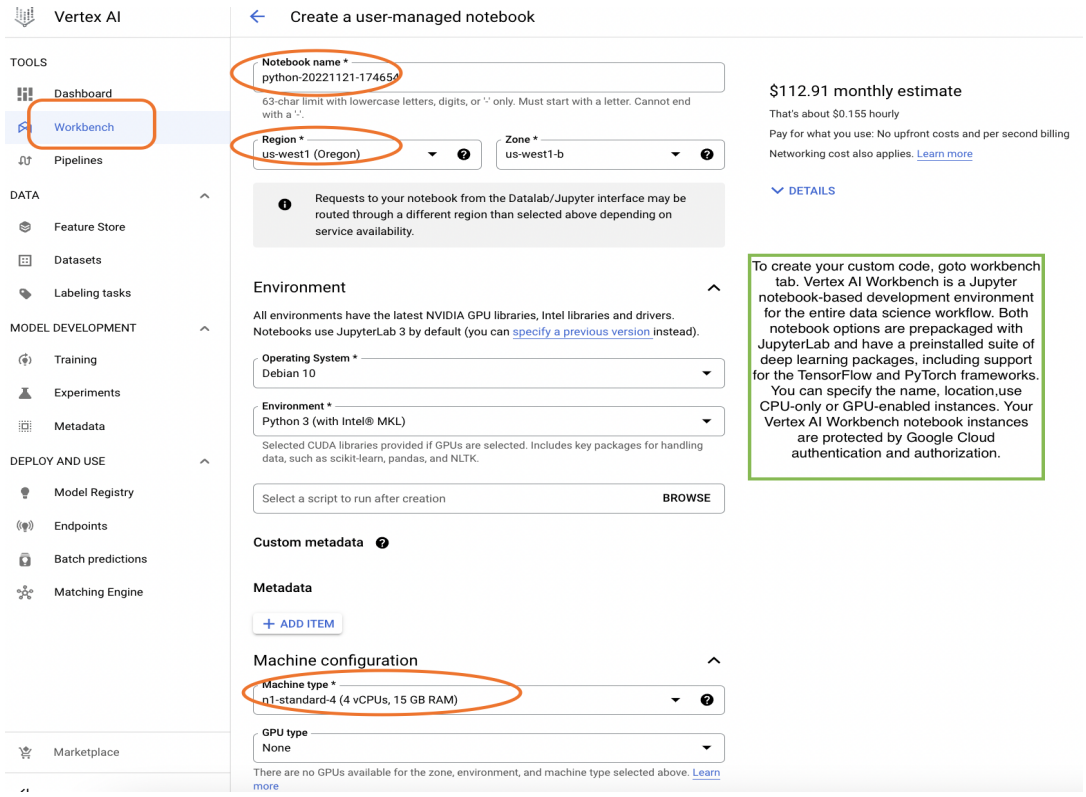


Image:[6]

Open Jupyter lab[7]

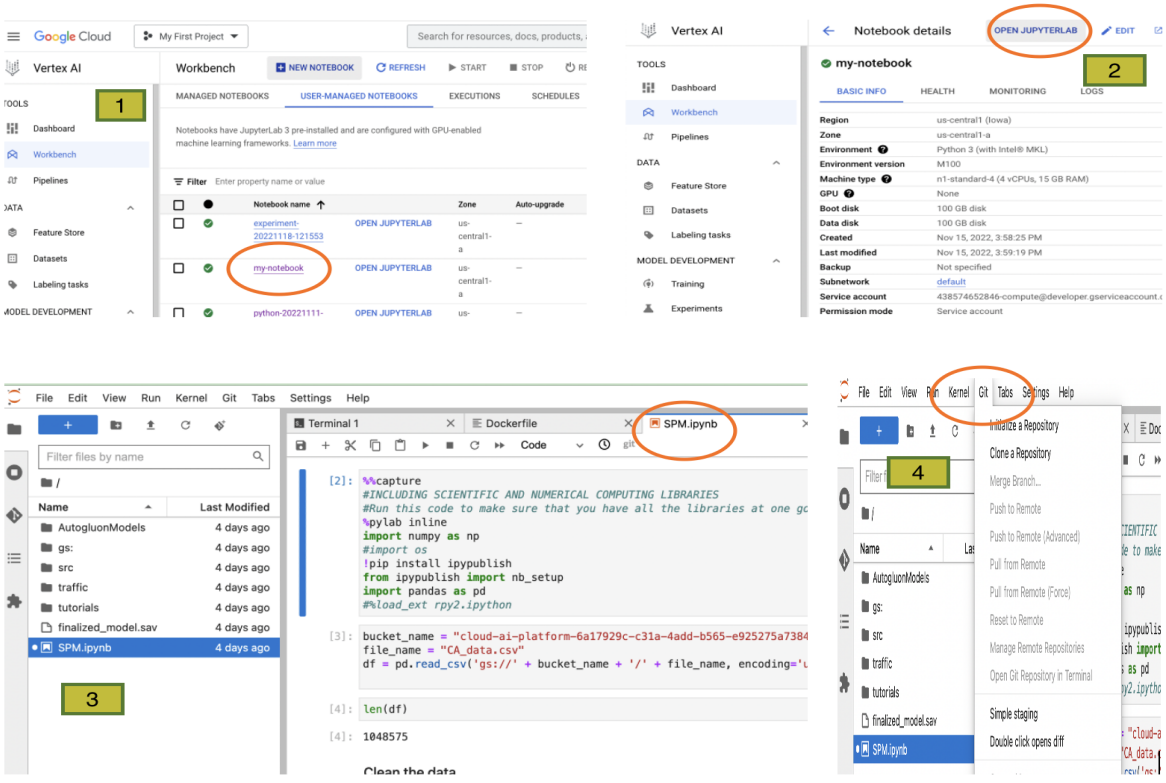


Image:[7]

Vertex AI Experiments tab[8]

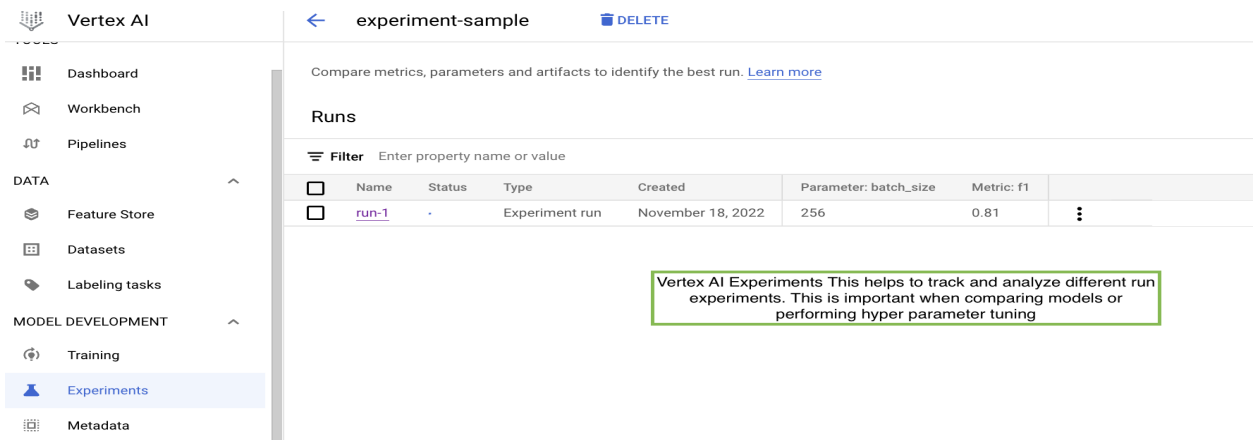


Image:[8]

Model Registry: Vertex Explainable AI on evaluation metrics[9]

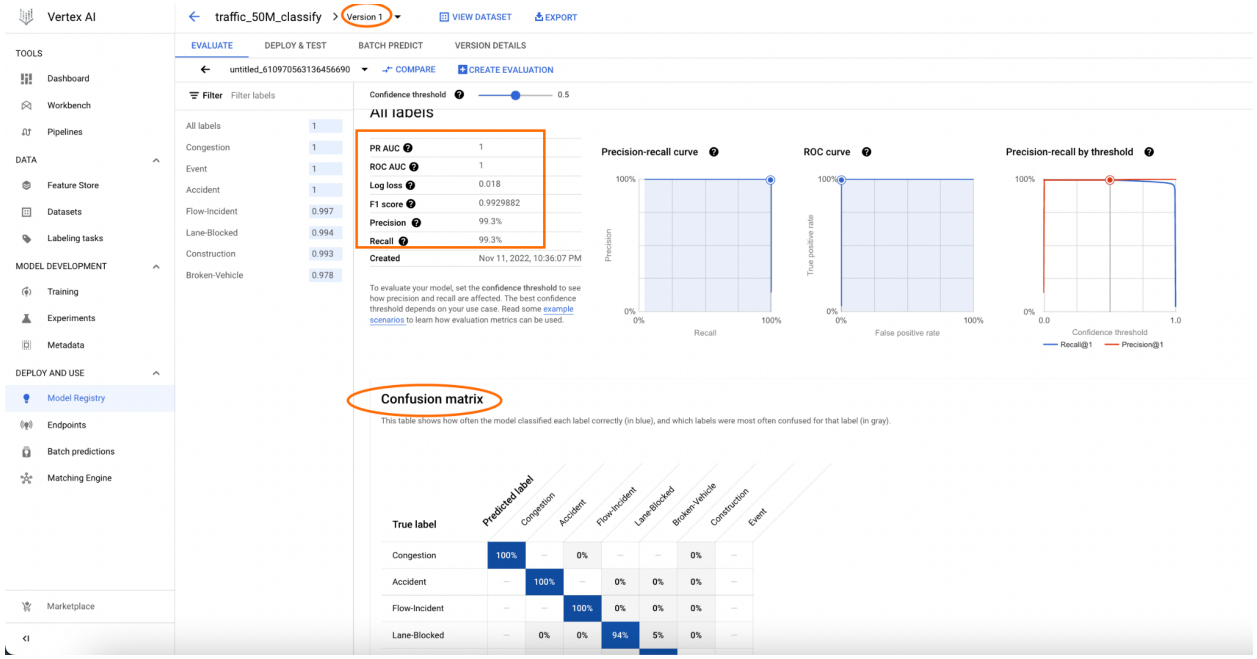
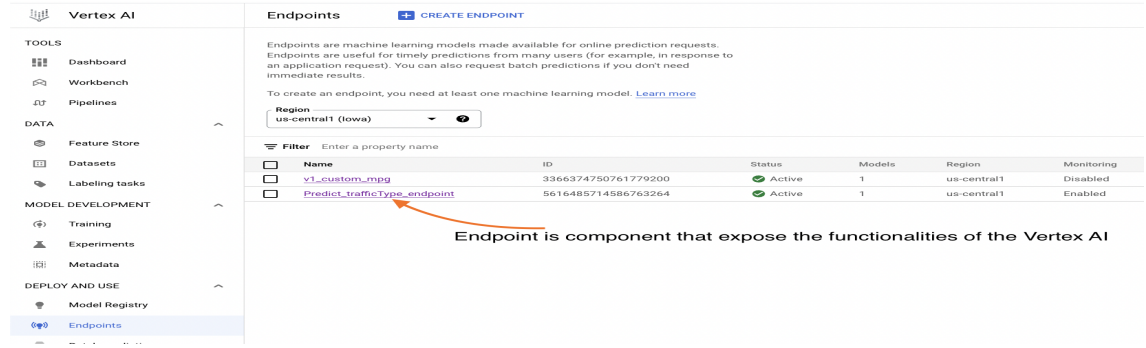


Image:[9]

C. Deployment

Endpoint [10]



Image[10]

MLFlow[6]

- MLflow is an open source platform for managing the end-to-end machine learning lifecycle
- Some benefits of MLFlow are Experiment Tracking, Model Management, and Model Deployment
- Four primary components of MLFlow are: Tracking, Projects, Models and Model Registry

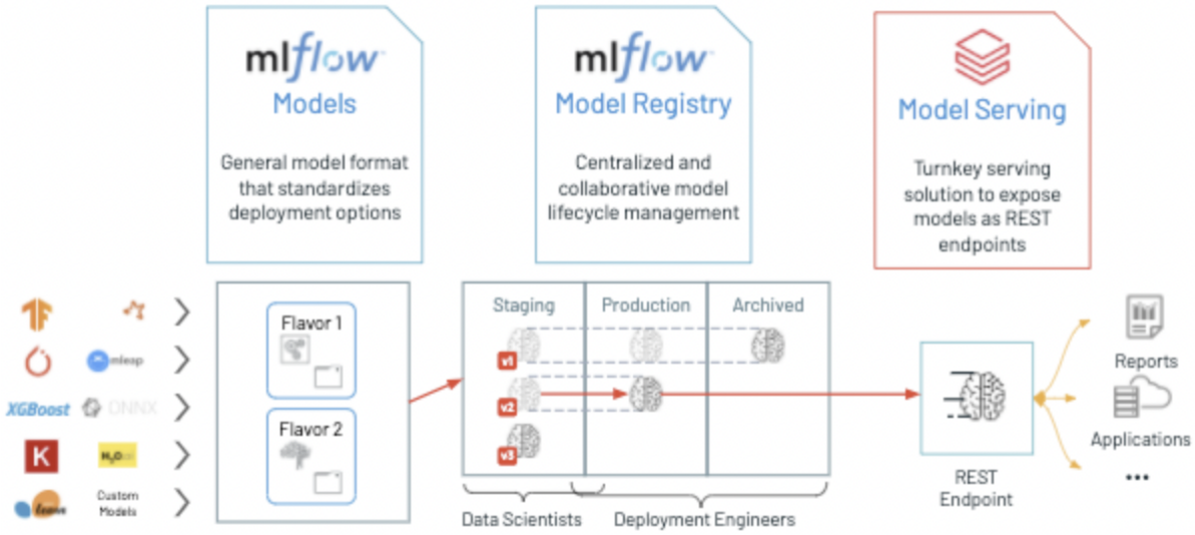


Figure 7 - MLFlow Serving (Image Credit)

MLFlow Components

Tracking (**25-30 hrs of manual work**)

- MLflow Tracking is organized around the concept of runs, which are executions of ML science code
- MLflow tracking provides an API and UI for logging parameters, code versions, metrics, and artifacts when running code and for later visualizing the results
- MLflow allows you to group runs under experiments, which can be useful for comparing runs intended to tackle a particular task
- You can record runs using MLflow Python, R, Java, and REST APIs from anywhere you run your code
- The MLFlow runs can be recorded either to local files, to a SQLAlchemy compatible database, or remotely to tracking server
- An MLFlow tracking server has two components: a backend store and an artifact store

- The backend store is where MLflow Tracking Server stores experiment and run metadata as well as parameters, metrics, and tags for runs
- The artifact store is a location suitable for large data (such as the GCS bucket) and is where you log their artifact output (for example, models)

Projects

- An MLflow Project is a format for packaging data science code in a reusable and reproducible way
- Each project is a directory of files, or a Git repository, containing your ML code. For example, projects can contain a conda.yaml file for specifying a Python Conda environment
- Each Project specifies properties such as: a human-readable *Name* of the project; *Entry Points* which include commands that can be run within the project and information about their parameters (calling .py file in project as entry point); *Environment* is the software environment that should be used to execute project entry points (includes all dependencies required by the project code)
- Software Environments supported by MLFlow Projects include Conda environments, Virtualenv environments, and Docker containers

Models (**2-4 days of manual work**)

- An MLflow Model is a standard format for packaging machine learning models that can be used in a variety of downstream tools (for e.g., real-time serving through a REST API or batch inference on Apache Spark)
- Offers a convention for packaging machine learning models in multiple flavors, and a variety of tools to help you deploy them
- MLflow provides tools to deploy many common model types to diverse platforms: for example, any model supporting the “Python function” flavor can be deployed to a Docker-based REST server, to cloud platforms such as Azure ML and AWS SageMaker, and as a user-defined function in Apache Spark for batch and streaming inference.

Registry **(~3 days of manual work)**

- Offers a centralized model store, set of APIs, and UI, to collaboratively manage the full lifecycle of an MLflow Model
- It provides model lineage (which MLflow experiment and run produced the model), model versioning, stage transitions (for example from staging to production or archiving), and annotations

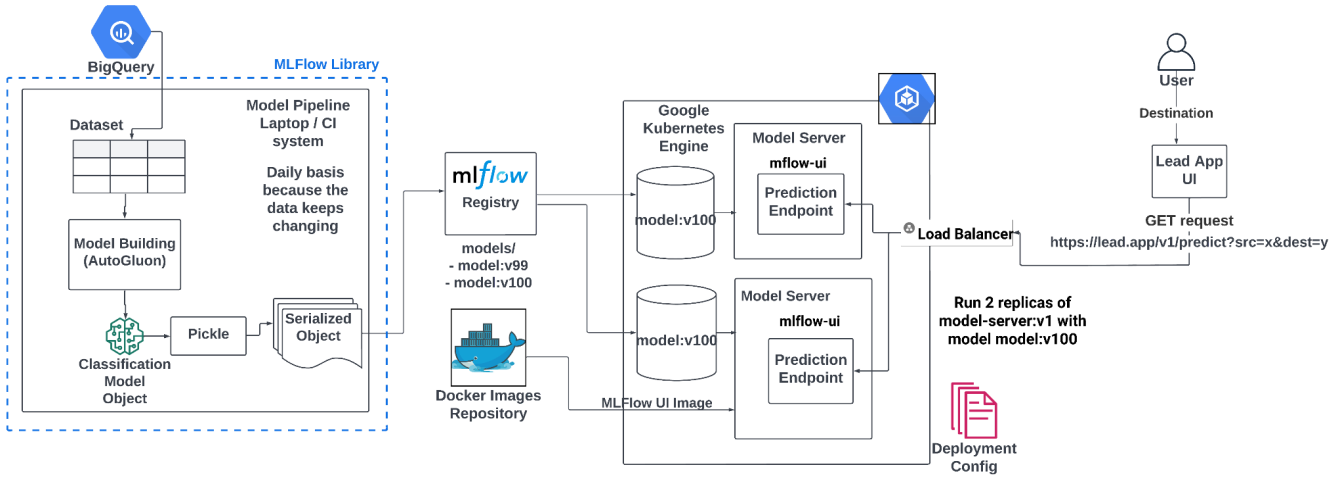


Figure 8 - MLFlow integration with the existing architecture

Advantages[7]

1. MLFlow is an Open Source MLOps tool and so all the features offered are free.
2. Offers model tracking, management, packaging, and centralized lifecycle stage transitions.
3. Aids in creating a collaborative environment for model development.
4. It is easy to adapt ML experiments to MLFlow as the tracking is done via a simple import in your code; It is a Python package that enables experiment tracking on current ML algorithms.
5. Experiment tracking, which is the core of MLFlow, favors the ability to develop locally and track runs in remote locations via a logging process. This is suitable for exploratory data analysis, model evaluation, and tuning hyperparameters in the ML Lifecycle.
6. MLFlow achieves model deployment by utilizing the Model Registry. Through this, MLFlow provides a central location to share ML models as well as a space for collaboration on how to move them forward for deployment.

7. Model Registry provides model versioning, model lineage, annotation, and stage transitions. It has a set of APIs and a UI to manage the complete lifecycle of the MLFlow model more collaboratively.
8. MLflow makes it easy to promote models on different cloud environments like Amazon Sagemaker, Microsoft Azure, and Google Cloud Platform.
9. Meets the needs of data scientists looking to organize themselves better around experiments and machine learning models. For such teams, ease of use and setup is often the main driver.
10. Saves time for data scientists as MLFlow provides features to visualize and compare experiments which contain various versions of the ML model performing a particular task (say, regression or classification).
11. The parameters for the ML model can be supplied to the pipeline as arguments which saves time as compared to re-running the entire Jupyter notebook. It also lets data scientists compare the performance of the model with changing parameters by logging all runs in the MLFlow Tracking server.
12. MLFLow provides out of the box APIs for Python, R, and Java. For any other language, it also provides REST APIs which allow the user to log runs, metrics, and parameters seamlessly.
13. MLflow has provided Documentation which gives in-depth information on all their components, how to install, and various samples to implement MLFlow.

Disadvantages[7]

1. MLFlow requires the data scientist to perform data cleaning and data pre-processing steps as well as model development from scratch rather than MLFLow providing an application to perform these steps; This requires the user to have prior ML model development knowledge/experience.
2. Does not automatically find a model that gives the best results (AutoML not supported).
3. Since AutoML is not supported, model development might take several days to weeks just to target a specific application for Lead.

4. MLFlow requires the current architecture for the deployment to be complete. MLFlow Recipes (also known as MLFlow Pipelines), a framework that enables data scientists to quickly develop high-quality models and deploy them to production, is still in the experimental phase.
5. The cost does not reduce much even though this is an open source tool as it requires the presence of current architecture for deployment and it still does not provide automation for ML model development unlike Google's Vertex AI.

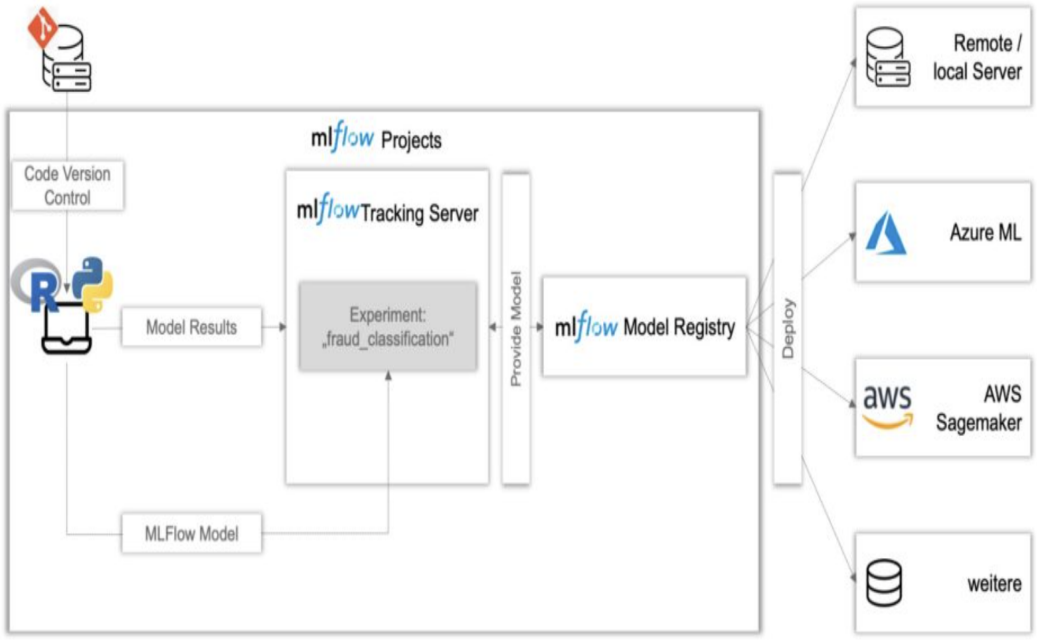


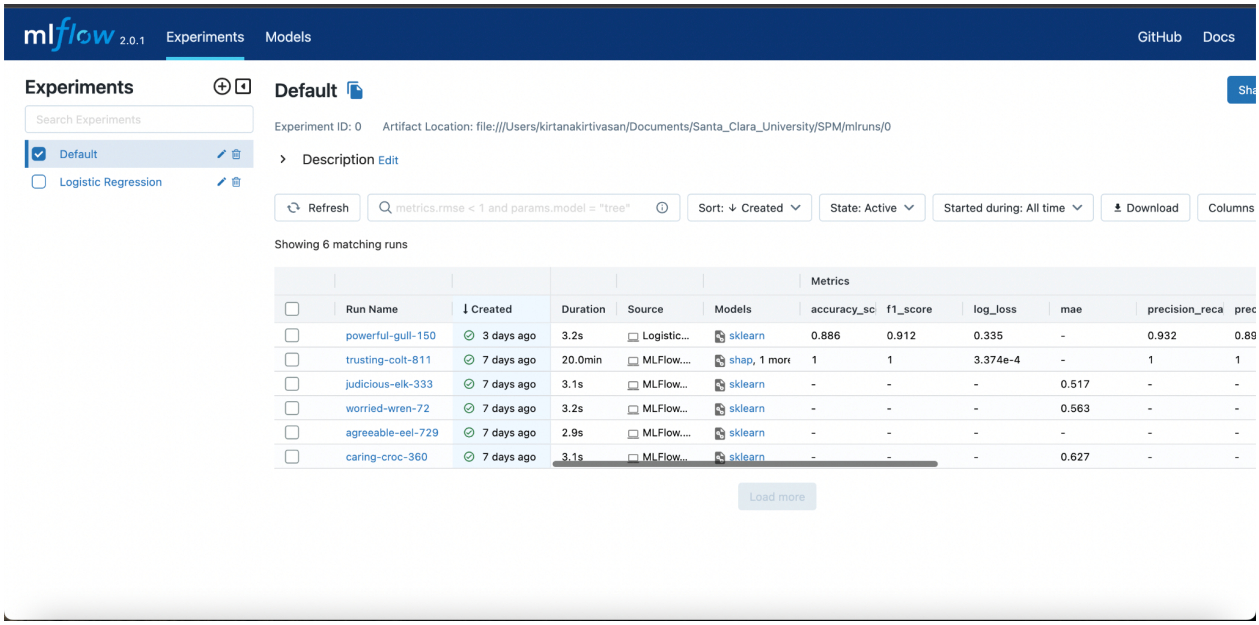
Figure 10 - Workflow Management with MLFlow (Image Credit)

MLFLOW DEMONSTRATION

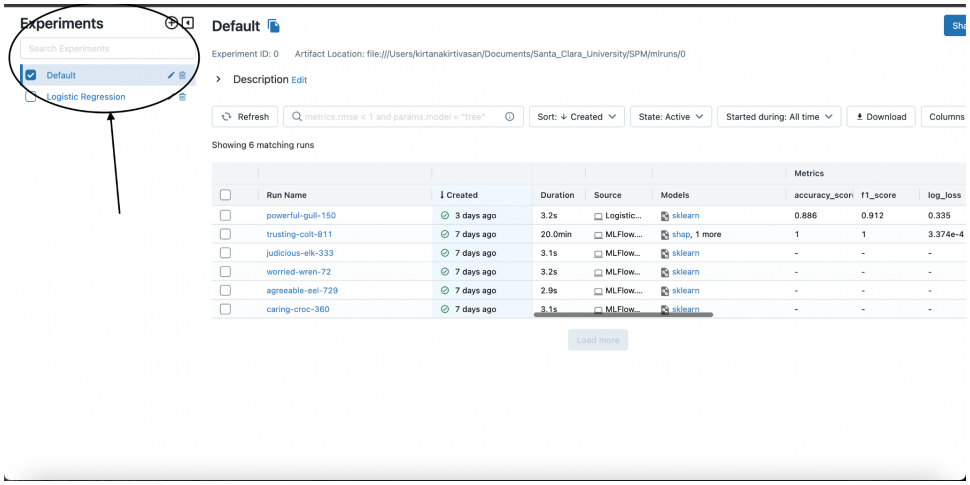
1. MLFlow Tracking API for starting and managing MLFlow runs

```
import mlflow
import mlflow.sklearn
import os

os.environ["MLFLOW_TRACKING_URI"]="http://127.0.0.1:5000/"

eval_data = X_test
eval_data["label"] = y_test

mlflow.set_experiment(experiment_id = '0')

with mlflow.start_run() as run:
    # Log the baseline model to MLflow
    mlflow.sklearn.log_model(model, "model")
    model_uri = mlflow.get_artifact_uri("model")

    # Evaluate the logged model
    result = mlflow.evaluate(
        model_uri,
        eval_data,
        targets="label",
        model_type="classifier",
        evaluators=["default"],
    )
```

2. MLFlow Tracking UI



3. MLFLow Experiments

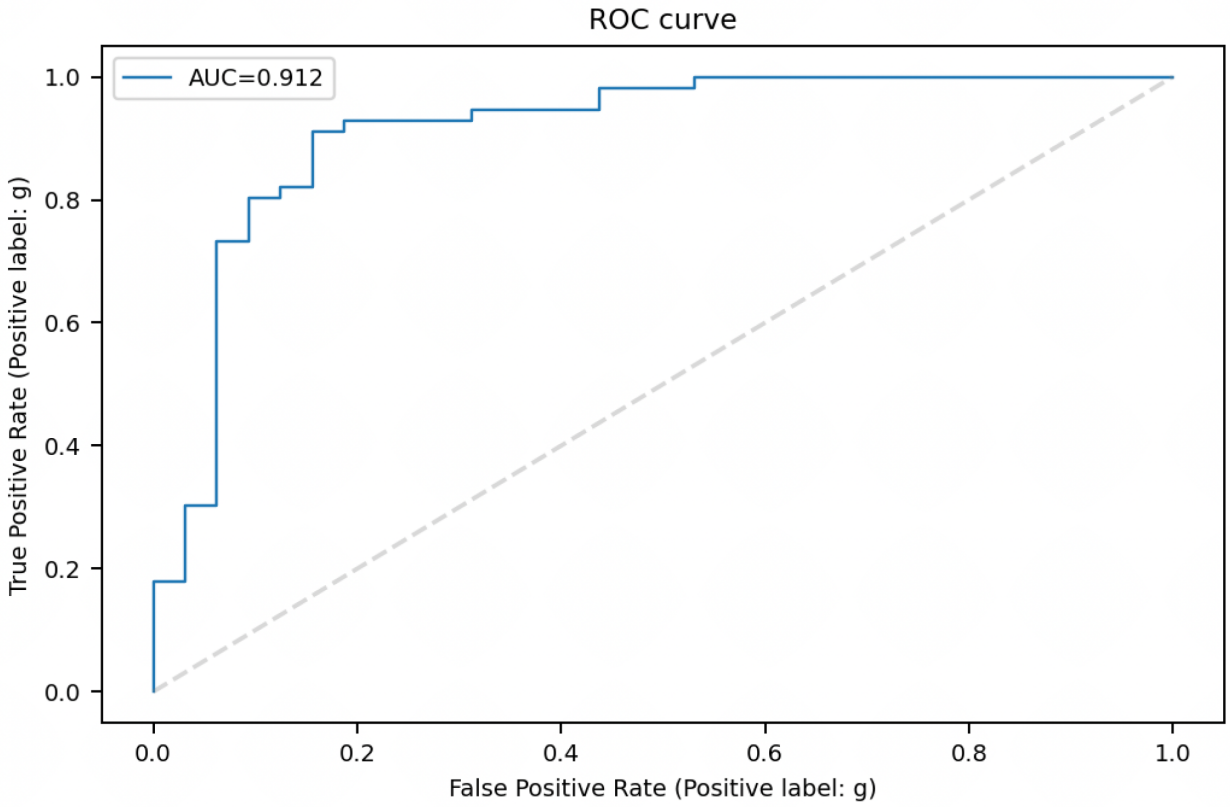


4. MLFlow Logged Metrics

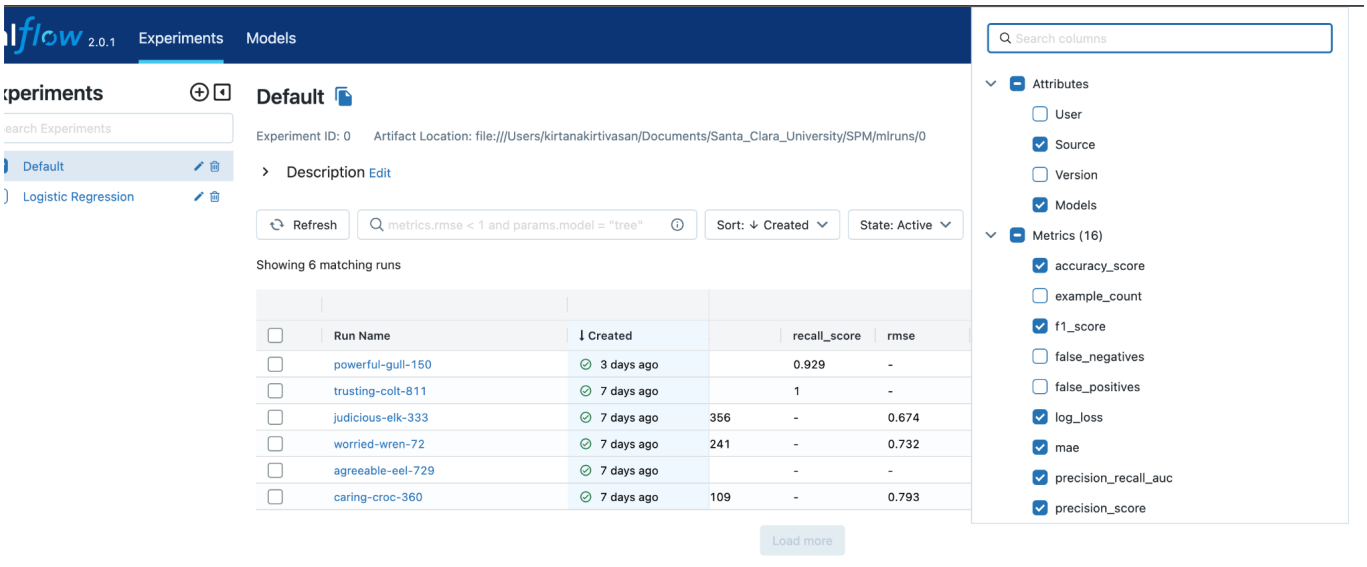
Metrics (13)

Name	Value
accuracy_score	0.886
example_count	88
f1_score	0.912
false_negatives	4
false_positives	6
log_loss	0.335
precision_recall_auc	0.932
precision_score	0.897
recall_score	0.929
roc_auc	0.912
score	0.886
true_negatives	26
true_positives	52

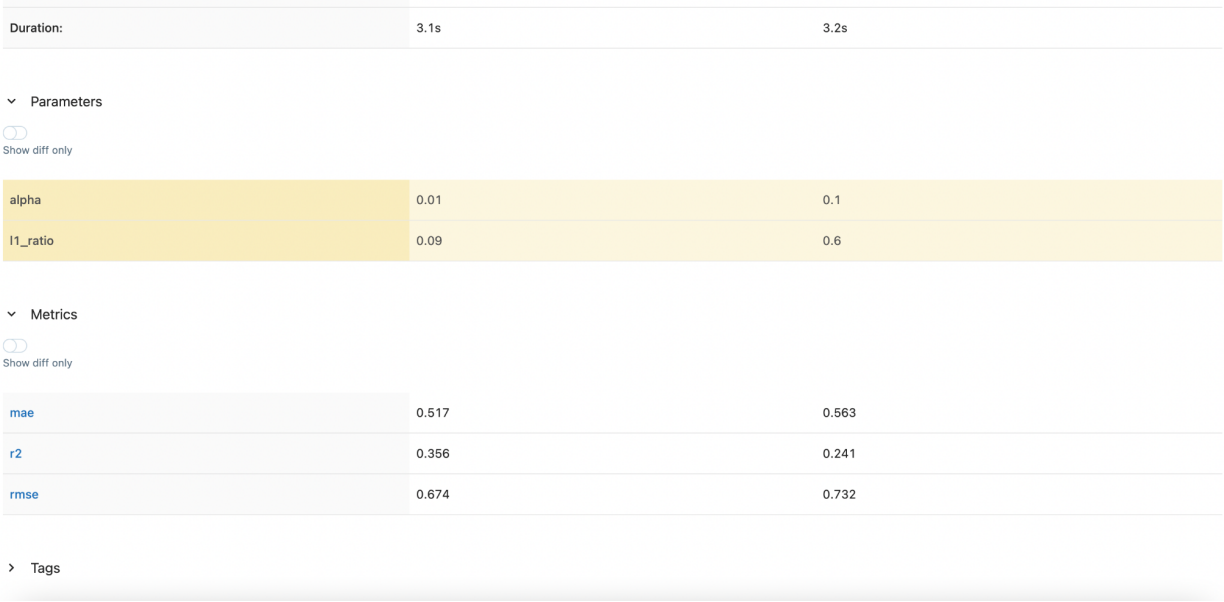
5. MLFlow Visualization



6. Filter Runs according to Metrics/Parameters/Attributes



7. Compare two Runs of same model (here, Linear Regression model sample) with different alpha and l1_ratio hyperparameters



Duration:	3.1s	3.2s

Parameters

Show diff only

alpha	0.01	0.1
l1_ratio	0.09	0.6

Metrics

Show diff only

mae	0.517	0.563
r2	0.356	0.241
rmse	0.674	0.732

Tags

Competitive Analysis

S.no	Feature	Vertex AI	MLflow
1	Python programming language support	Yes	Yes
2	R programming language support	Yes	Yes
3	Read any data source using SQL syntax	Metadata only	Uses Hive metastore
4	Ability to save, load, list, tag and version multiple models	Yes (training models and model server versions)	Yes (training models version)
5	Store metrics and details of ML model training	Yes	Yes (MLFlow Experiments)
6	Feature store - Save pre calculated tabular data to be used by other team members	Yes	No
7	Scheduling - Run notebooks, jobs or pipelines on regular intervals	Requires cloud scheduler	No
8	Integration	Natively integrated with BigQuery, Dataproc and Spark	Plugins allow to integrate with third-party storage solutions for experiments data (metrics and hyperparameters), artifacts and models
9	API Abstraction with simple one line code or User interface - Unified UI and API	Yes	Yes
10	AutoML - Try to automatically find a model that gives the best results	Yes	No

11	User Interface support for ML workflow	Supports end to end - development, serving, monitoring	Supports only for tracking and has its own registry
12	Support for open source frameworks - tensorflow, pytorch, scikit-learn	Yes	Yes
13	Use cases	● Data readiness ● Feature Engineering ● Training & hyperparameter tuning ● Model serving ● Model tuning and understanding ● Model monitoring ● Model management	● Tracking ● Projects ● Models ● Registry
14	Time reduction through MLOps automation	~ 15 days of manual labor	~ 6-8 days of manual work

COST COMPARISON

Vertex AI

Training[5]

- 8 hours per model
- Per hour pricing
    - n1-standard-8, 8 CPU, 32 GB Memory = $0.436998 / hr
    - NVIDIA_TESLA_K80 GPU = $0.517500 / hr
    - ~$1 per hour = ~$8 per day per model
    - 8*365 = $3000 per year per model

Serving[5]

Per hour pricing

- n2-standard-8 $0.39 per hour
- 1 machine $0.39 per hour = $3,416 per year (if running all the time)
- Scale up to 10 machines during a hypothetical two-hour peak that occurs everyday
    - 9 machines $0.39 per hour = $2,562 per year (2 hrs per day)
- Total = $6,000

Engineering / deployment

- Initial CI/CD pipeline setup cost = $6,000

Costs for deploying 1 model to production

Model Infrastructure	Kubernetes clusters with a load balancer	$6,000
Data Support	Independent data pipeline manager for continuous update of analytic data	$10,000 (labor) + $3,000
Engineering / Deployment	Continuous integration and continuous deployment (CI/CD) system to pull model from registry	$6,000 (labor) + $200
Total Investment / yr		$16,000(labor) + $9200
5 year TCO		$62,000

Costs for deploying an additional model to production

Model Infrastructure	Cluster can be adjusted to run multiple models according to changing using demands	+ $2,561 / yr
Data Support	Modules can be added to pipeline management system	+ $2,250 (labor)
Engineering / Deployment	CI/CD system can be extended to pick up additional model	+ $750 (labor)
Total Investment / yr		$3,000(labor) + $2,561 / yr
5 year TCO		+ $15,805

1. Cost for deploying 1 model to production = $62,000 over first 5 years
2. Cost for deploying 2 models to production = $77,805 over first 5 years
3. Cost for deploying 100 models to production = $1.63 Million over first 5 years

MLFlow

Training

- Same as existing architecture

Serving

- Same as existing architecture

Engineering/deployment

- Initial CI/CD pipeline setup cost = $18,000

Costs for deploying 1 model to production

Model Infrastructure	Kubernetes clusters with a load balancer	$7,000
Data Support	Independent data pipeline manager for continuous update of analytic data	$7,000 (labor) + $3,200
Engineering / Deployment	Continuous integration and continuous deployment (CI/CD) system to pull model from registry	$18,000 (labor) + $300
Total Investment / yr		$25,000(labor) + $11,500
5 year TCO		$81,000

Costs for deploying an additional model to production

Model Infrastructure	Cluster can be adjusted to run multiple models according to changing using demands	+ $3,781 / yr
Data Support	Modules can be added to pipeline management system	+ $1,575 (labor)
Engineering / Deployment	CI/CD system can be extended to pick up additional model	+ $2,250 (labor)
Total Investment / yr		$3,825(labor) + $3,781 / yr
5 year TCO		+ $22,730

1. Cost for deploying 1 model to production = $81,000 over first 5 years
2. Cost for deploying 2 models to production = $103,730 over first 5 years
1. Cost for deploying 100 models to production = $2.3 Million over first 5 years

RECOMMENDATION

The best alternative is **Vertex AI** because,

- Vertex AI provides a **unified platform** that can accommodate data readiness to ML model deployment. MLFlow doesn't streamline the entire ML lifecycle.
- With Vertex AI, a single Data Scientist could close the product cycle from research to production, which **saves the labor cost and time** for the entire lifecycle. MLFlow only partially streamlines the deployment process and uses the current system infrastructure for deployment. So the time taken and operational/labor cost is just a little lesser than the current system despite being open source.
- Vertex AI natively uses Google infrastructure and our system is majorly on Google Cloud. So the adoption of Vertex AI to our current system is more **compatible**.

PROCESS MODEL & PLANNING ACTIVITIES

Project Phases

This is a ‘software purchase and adoption’ project to streamline the ML model deployment. The project can be divided into 2 main phases.

1. Research
    a. Research on ML Operations
    b. Identify a data science specific use case
    c. Research on the working of the current system in the context of the use case
    d. Identify the problems with the current system
    e. Requirements gathering
    f. Research on 2 managed services evaluating different factors
    g. Identify the best managed service for our project and prepare guidelines on how it can be adopted into our system.
    h. Produce final report
2. Implementation of the adopted system (Demo)

    This phase is an end-to-end Data Science project by itself. The phases included are,

    1. Data understanding
    2. Data preparation
    3. Modeling
    4. Evaluation
    5. Deployment

We have 2 phases with different functionalities. So we need to identify a model that works best for both documentation heavy tasks and the implementation of a data science project by itself.

Choice of Process Model

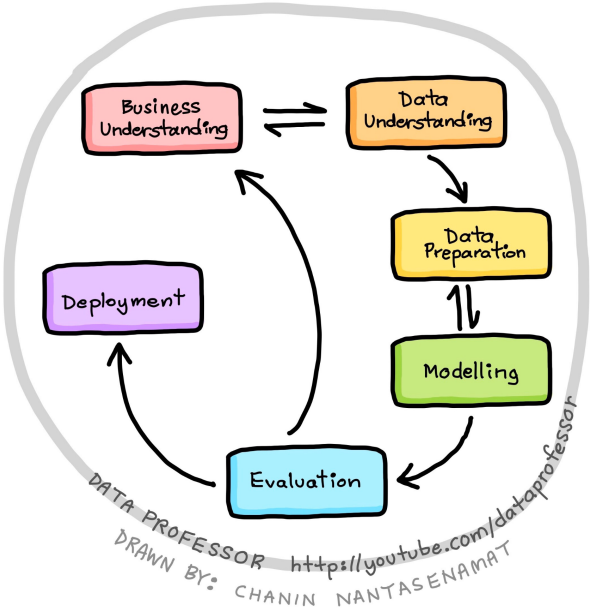


Figure 11 - CRISP-DM Model (Image Credit)

CRISP-DM works best for the project since,

1. It is generalizable among Data Science projects
2. A 'mix & match' model which accommodates both horizontal and vertical slicing, i.e, waterfall and agile
3. In our project, the research phase follows a horizontal slicing which helps the team to move to the implementation phase with a clear understanding of the business needs
4. The implementation phase follows a vertical slicing which helps the team to deliver frequently and keep improving the model
5. Small team size (4 members)

PROJECT PLANNING & TRACKING

CRISP-DM Horizontal Slicing (Waterfall)[8]

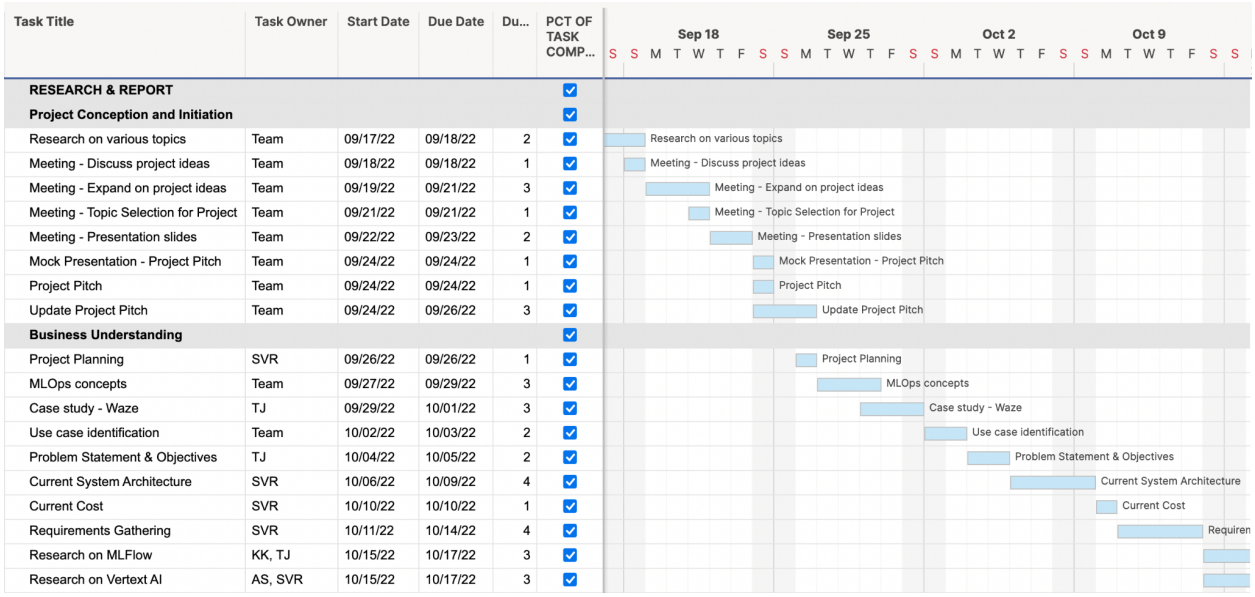
The research phase of the project includes,

1. Research on managed services and MLOps topics for the project
2. Case study of Waze
3. Research on current system architecture, cost and problems
4. Report writing

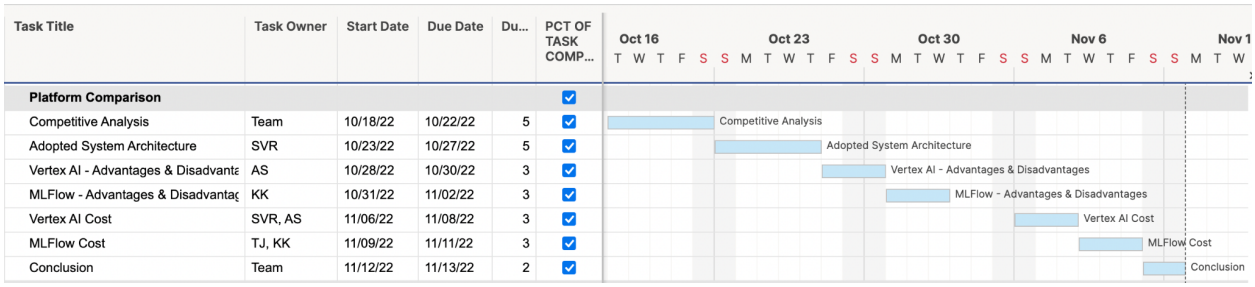
In the research phase of the project,

- Upcoming tasks are dependent on the results from the previous tasks
- Heavy emphasis on documentation was followed

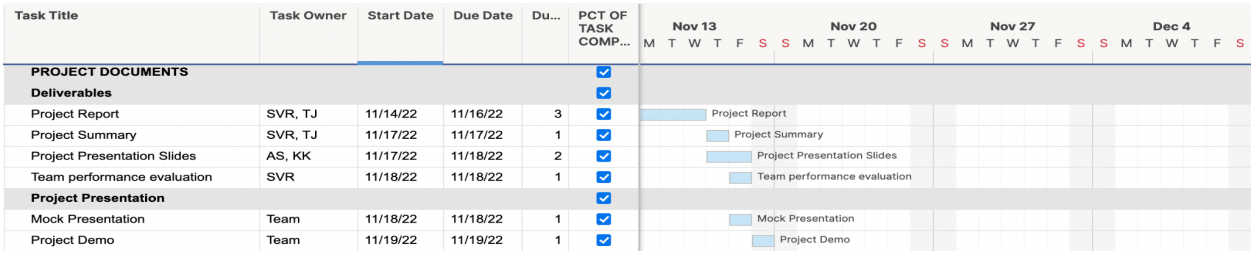
Considering the above factors, a horizontal slicing(waterfall) in the CRISP-DM model works best.



Task Title	Task Owner	Start Date	Due Date	Du...	PCT OF TASK COMP...
RESEARCH & REPORT					☑
Project Conception and Initiation					☑
Research on various topics	Team	09/17/22	09/18/22	2	☑
Meeting - Discuss project ideas	Team	09/18/22	09/18/22	1	☑
Meeting - Expand on project ideas	Team	09/19/22	09/21/22	3	☑
Meeting - Topic Selection for Project	Team	09/21/22	09/21/22	1	☑
Meeting - Presentation slides	Team	09/22/22	09/23/22	2	☑
Mock Presentation - Project Pitch	Team	09/24/22	09/24/22	1	☑
Project Pitch	Team	09/24/22	09/24/22	1	☑
Update Project Pitch	Team	09/24/22	09/26/22	3	☑
Business Understanding					☑
Project Planning	SVR	09/26/22	09/26/22	1	☑
MLOps concepts	Team	09/27/22	09/29/22	3	☑
Case study - Waze	TJ	09/29/22	10/01/22	3	☑
Use case identification	Team	10/02/22	10/03/22	2	☑
Problem Statement & Objectives	TJ	10/04/22	10/05/22	2	☑
Current System Architecture	SVR	10/06/22	10/09/22	4	☑
Current Cost	SVR	10/10/22	10/10/22	1	☑
Requirements Gathering	SVR	10/11/22	10/14/22	4	☑
Research on MLFlow	KK, TJ	10/15/22	10/17/22	3	☑
Research on Vertext AI	AS, SVR	10/15/22	10/17/22	3	☑



Task Title	Task Owner	Start Date	Due Date	Du...	PCT OF TASK COMP...
Platform Comparison					☑
Competitive Analysis	Team	10/18/22	10/22/22	5	☑
Adopted System Architecture	SVR	10/23/22	10/27/22	5	☑
Vertex AI - Advantages & Disadvanta	AS	10/28/22	10/30/22	3	☑
MLFlow - Advantages & Disadvantag	KK	10/31/22	11/02/22	3	☑
Vertex AI Cost	SVR, AS	11/06/22	11/08/22	3	☑
MLFlow Cost	TJ, KK	11/09/22	11/11/22	3	☑
Conclusion	Team	11/12/22	11/13/22	2	☑



CRISP-DM Vertical Slicing (Agile)[8]

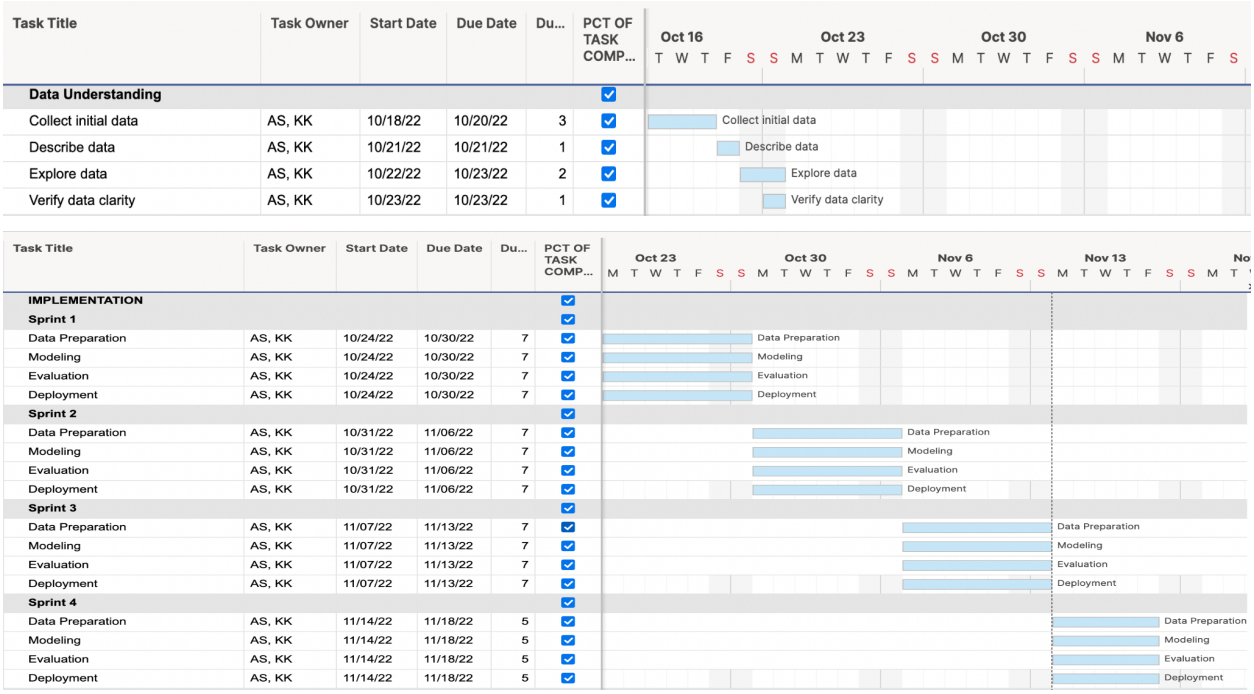
The implementation phase of the project includes,

1. Working demo on Vertex AI
2. Working demo on MLFlow

In this phase,

- The team had a clarity on the project requirements from the research phase.
- The team worked on the demo iteratively with heavy emphasis on construction activities.
- The demos were optimized in the future sprints
- The team had a 15 mins standup meetings daily to update the demo status
- A portion of the overall demo was expected to be implemented at the end of each week

Considering the above factors, a vertical slicing(agile) in the CRISP-DM model works best.



REFERENCES

[1] Under the Hood: Real-time ETA and How Waze Knows You're on the Fastest Route

[2] The Life Cycle of a Machine Learning Project: What Are the Stages? - neptune.ai

[3] Estimate the Time, Cost, and Deliverables of an ML App Project

[4] What is the Cost to Deploy and Maintain a Machine Learning Model

[5] Vertex AI Pricing

[6] MLFlow Documentation

[7] A Comprehensive Comparison Between Kubeflow and MLflow

[8] What is CRISP DM?

[9] Vertex AI