

15-319 / 15-619

Cloud Computing

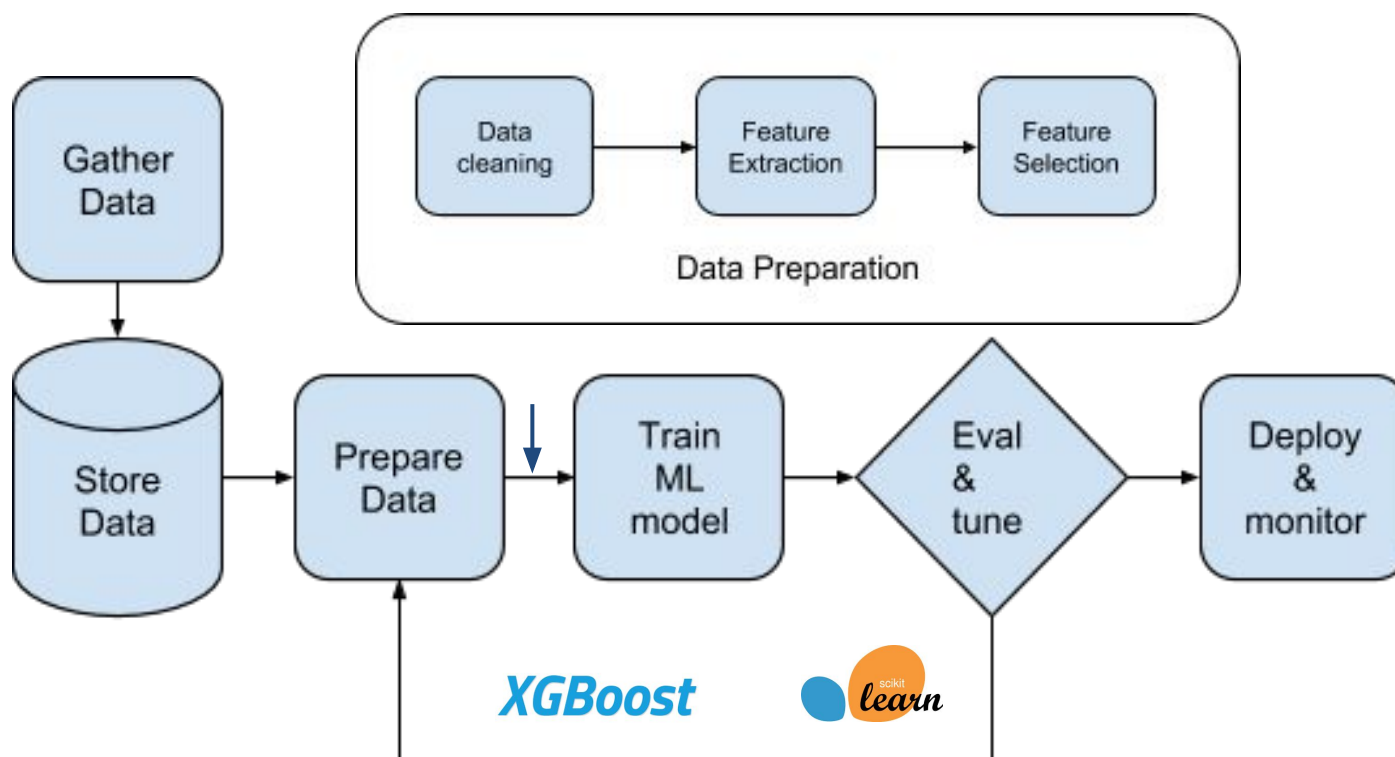
Overview 12

April 20th 2021

Overview

- **Last week's reflection**
 - Team Project Phase 2, Live Test
 - Quiz 10
- **This week's schedule**
 - Unit 5 - Modules 21, 22
 - Quiz 11
 - Project 4.2
 - Machine Learning on the cloud
 - **Twitter Analytics: The Team Project**
 - Phase 3
 - Managed Services

Machine Learning in Production



ML on Managed Services

- Model training on large datasets tends to be computationally intensive
- An increasingly affordable option for users without specialized IT infrastructure is to process ML workloads on the cloud with Managed Services like the Google AI Platform.
- Benefits:
 - No need to provision and configure virtual machines
 - Horizontal and vertical scaling is possible
 - No need to write custom logic to orchestrate multiple workers and achieve parallel training
 - Deploy your model to the cloud

P4.2 - Taxi Fare Prediction Application

- Accepts speech queries with origin and destination, returns fare estimate as speech



I would like to get from Central Park Zoo to Grand Central Terminal



Your expected fare from Central Park Zoo to Grand Central Terminal is \$29.69

P4.2 - Overview of Tasks

- Task 1: Data Visualization and Feature Engineering
- Task 2: Training, hyperparam tuning, deploying your model using the Google AI Platform and serving queries.
- Task 3: Stitch together services into a pipeline to build a user-facing interface for fare predictions.
- Bonus:
 - Use Cloud Vision API to identify NYC landmarks
 - Use AutoML to train a custom model that accepts custom landmarks as input for prediction

Task 1: Feature Engineering

- Data Visualization

- You are given a small training dataset containing historical data of fare prices in New York City.
- Steps to perform
 - Data exploration and visualization
 - Understand the data for Feature Engineering with regards to feature construction, data cleaning, etc.

Task 1: Feature Engineering

- You are given a small training dataset containing historical data of fare prices in New York City
- Steps to perform
 - Clean the data and remove outliers
 - Consider what you learned from the data visualization task
 - Extract or construct meaningful features that will improve performance over the baseline model (which uses raw features with no transformations)

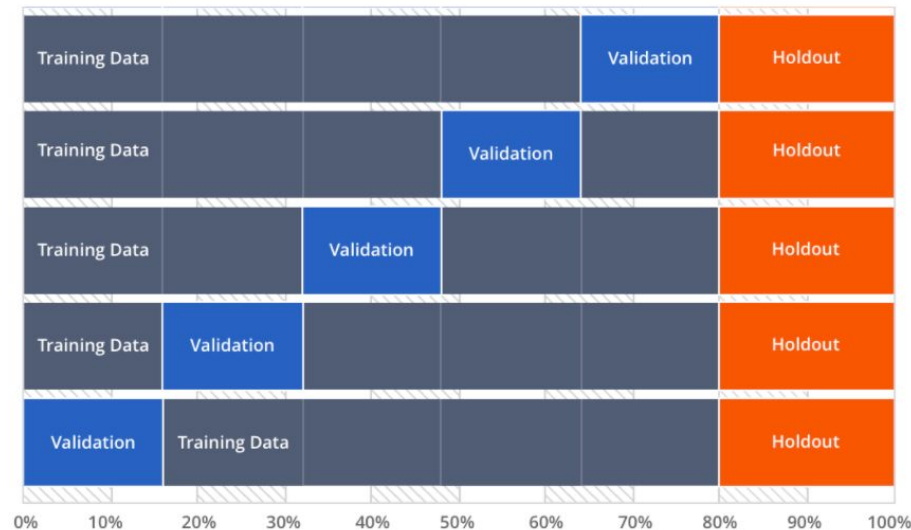
Task 1: Feature Engineering

- Feature engineering = transforming domain knowledge into better features
- Some ideas for feature engineering
 - Calculate distance from the geo-coordinates
 - Calculate distance to landmarks
 - What are good proxies for traffic conditions?



Task 1: Feature Engineering

- Evaluating your model
 - Metric: Root Mean Squared Error (RMSE)
 - K-fold Cross-Validation
 - Used to assess the predictive performance of the model outside the training sample on unseen data



- Plot feature importance

Task 2: Training, Tuning & Deploying

- Train and tune the model on complete dataset on Google AI Platform.
- Deploy the trained model to AI Platform.
- Develop a Flask application that accepts web requests and returns fare predictions.
 - Transform raw features from web requests using the feature engineering solution developed in Task 1.
 - Make API calls to the model hosted on AI Platform
 - Format and return a web response
- Deploy the Flask application on Google App Engine.

Task 2: Tuning with Google AI Platform

- **Hyperparameter Tuning**
- Parameters v/s. Hyperparameters
 - Parameters: internal, often not set by the practitioners
 - Hyperparameters: external, often set by the practitioners before training
 - Basically, configuration parameters that impact the training process
- Finding optimal hyperparameters with an exhaustive Grid Search is expensive

Task 2: Tuning with GCP HyperTune

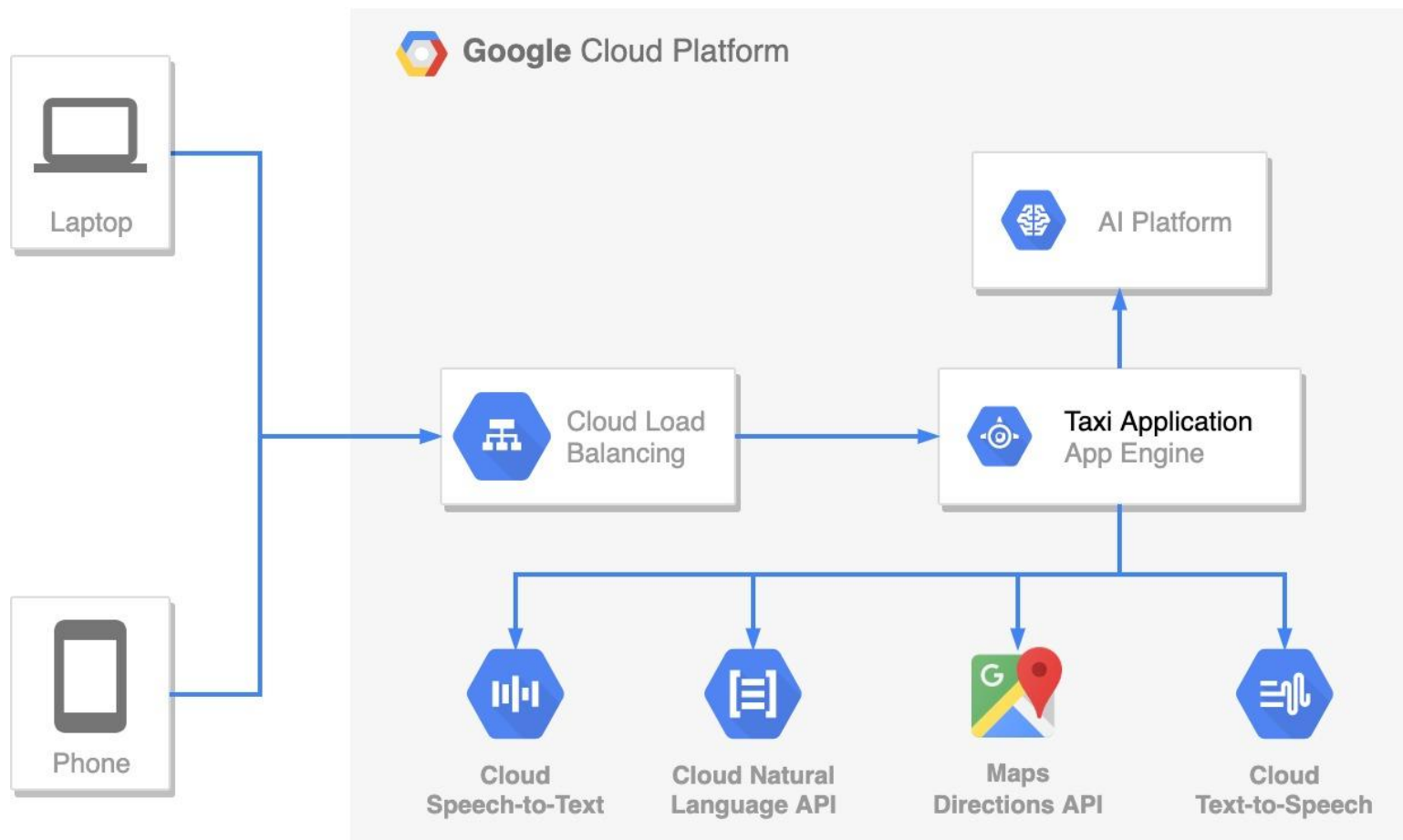
- Black box optimization service (does not need access to the underlying model)
- Need to specify a config yaml file that describes which hyperparameters to tune
- Uses a method called Bayesian Optimization to efficiently search through different combinations of hyperparameters

Task 2: Deploying Model to AI Platform

- **To get a full score in this task, you need to:**
 - Enable HyperTune, add at least 3 additional parameters to tune, run the hypertuning job and create a model on Google AI Platform.
 - Deploy the fare prediction application to GAE that uses the model you created and serves web requests correctly.
 - The predictions should achieve a target accuracy, measured by RMSE.

Task 3: ML Application Pipeline

- Build an end-to-end application pipeline to predict car fare requests using the following architecture.



Task 3: ML Application Pipeline

- Your application will include multiple APIs
 - Functional APIs to be implemented
 - **/predict** - Generate fare predictions for a JSON array of inputs
 - **/speechToText** - Convert WAV audio to text string
 - **/textToSpeech** - Convert text string to WAV audio
 - **/namedEntities** - Identify landmarks in a given sentence
 - **/directions** - For two given NYC landmarks, determine the latitude / longitude for each pickup and drop off pair

Task 3: ML Application Pipeline

Putting it together:

- **/farePrediction** - Given a WAV audio ride request, determine the predicted fare
 - **Response**
 - { "predicted_fare": "23.78",
"entities": ["Charging Bull", "Carnegie Hall"],
"text": "Your expected fare from Charging Bull to Carnegie Hall is \$23.78",
"speech": <BASE64 ENCODED AUDIO> }
- General solution flow
 - Speech to text ride request (/speechToText)
 - Extract entities from text ride request (/namedEntities)
 - Get the coordinates of the pickup and drop off locations (/directions)
 - Query the AI Platform model to get the predicted fare (/predict)
 - Convert the text response to speech (/textToSpeech)

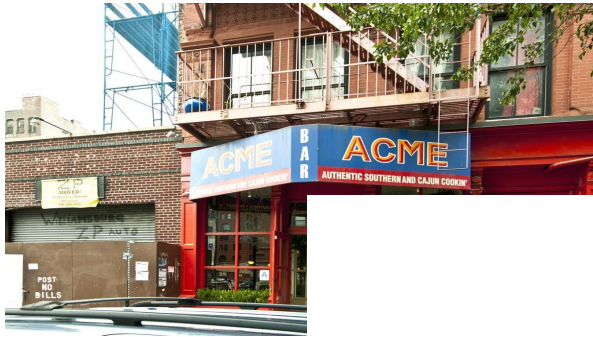
Bonus: Landmark Recognition

- (5 points) Use Cloud Vision to identify NYC landmarks
- (5 points) Add unique destinations using AutoML
- **/farePredictionVision**
 - Unlike **/farePrediction**, the ride request will not be sent as WAV audio
 - The API will accept the source and destination as images of NYC landmarks
 - Must query the Cloud Vision API and custom AutoML model to determine the landmark names
 - Continue with the same request as **/farePrediction**

Bonus: Landmark Recognition



Bonus: Landmark Recognition



Cloud AutoML Vision



Hints

- Task 1: Feature transformation
 - The exact same feature transformations must be applied to the training and the test set
 - Cannot share code if stateful functions are used, for example:
 - `get_dummies()`
 - `df.qcut()`
 - Store state like bin ranges and categorical values to apply the transformation consistently
 - Jupyter: command not found (use `virtualenv`)

Hints

- Task 2: HyperTune
 - Read the XGBoost hyperparameter doc to understand which hyperparameters can help most.
 - You can change the number of workers for the AI Platform training job to parallelize the training process.
 - Learn to make good estimates for the cost for each run
 - $\text{Cost} = \text{Consumed ML Units} * \0.49

Issues to Consider

- Overfitting
 - RMSE on training data is much lower than test data
 - You should not filter outliers just because it makes your cross validation scores look better, since some of these records may be representative of the patterns in the real world.
 - Students who do this may pass Task 1, but will fail Task 2.
 - You should make sure you have good features first, before trying to play around with filtering outliers.

Upcoming Deadlines

- Team Project: Phase 2
 - Code and report due:
 - Tuesday, Apr 20, 2021 11:59 PM ET
- Project 4.2: Machine Learning on the Cloud
 - Due Sunday, Apr 25, 2021, 11:59 PM ET
- Team Project: Phase 3
 - Live-test:
 - Sunday, May 02, 2021 6:00 PM ET
 - Code and report due:
 - Tuesday, May 04, 2021 11:59 PM ET