15-319 / 15-619
Cloud Computing

Recitation 13

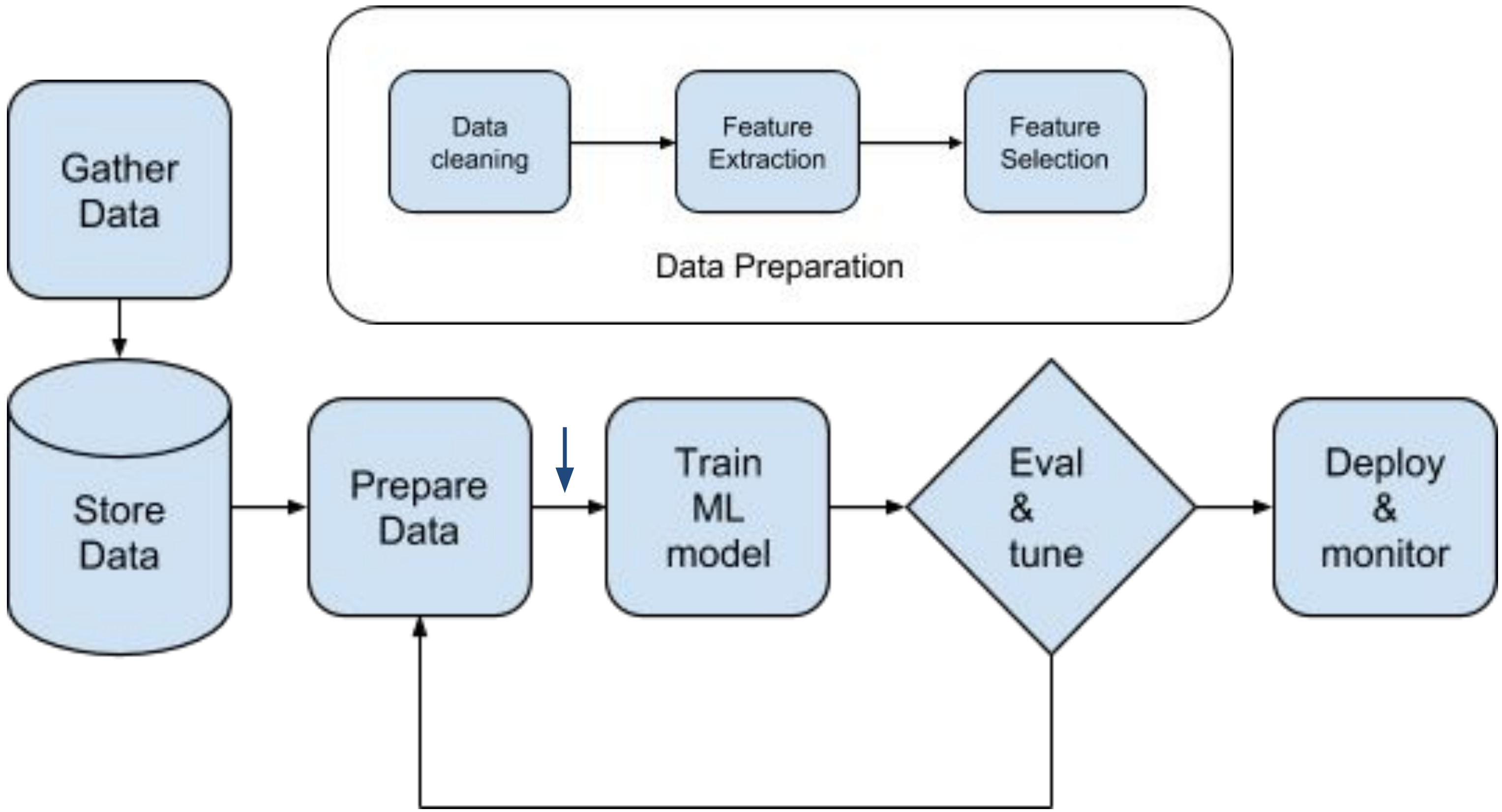
April 16th 2019

Overview

- **Last week's reflection**
  - Team Project Phase 2, Live Test
  - Quiz 11
- **This week's schedule**
  - Project 4.2
  - **Twitter Analytics: The Team Project**
    - Phase 3
      - Managed Services

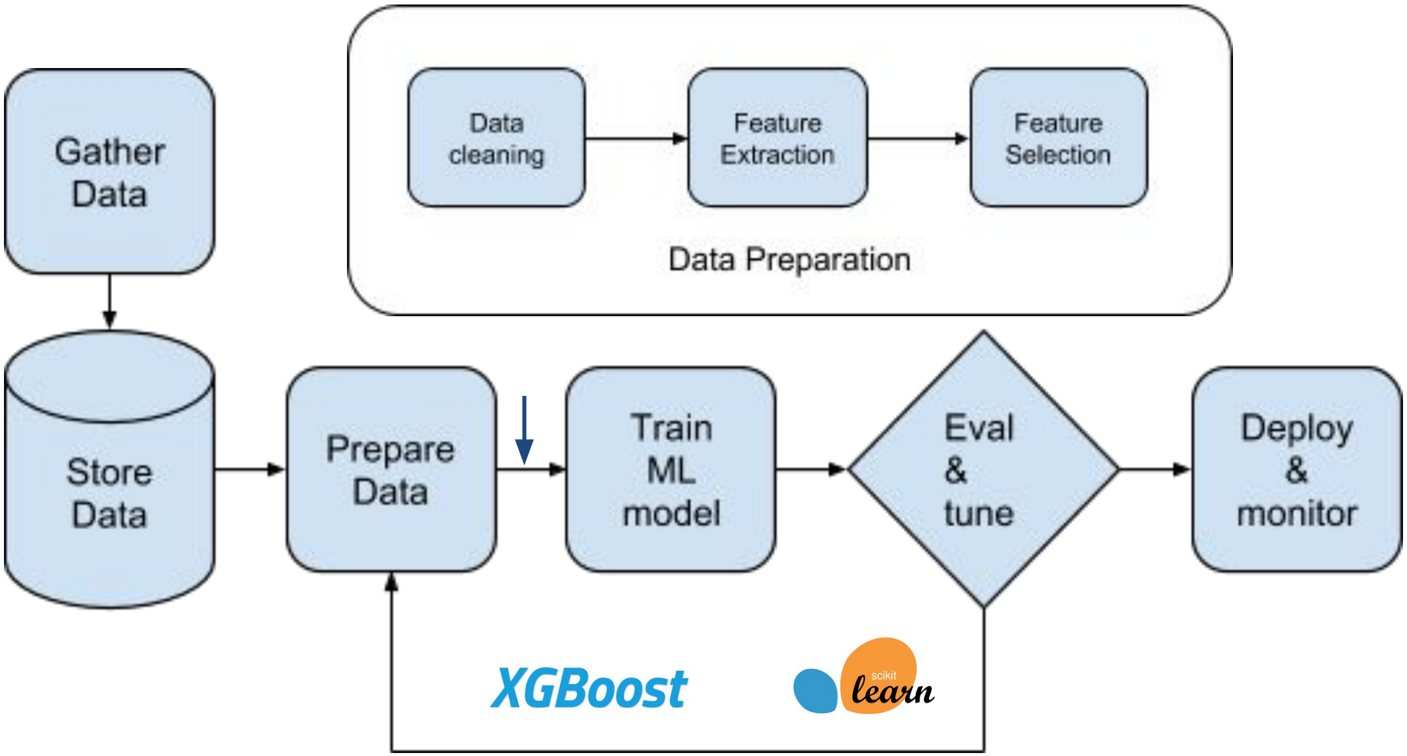
Machine Learning in Production

- A typical end-to-end process for Machine Learning



Machine Learning in Production

- A proliferation of tools on the Cloud



jupyter

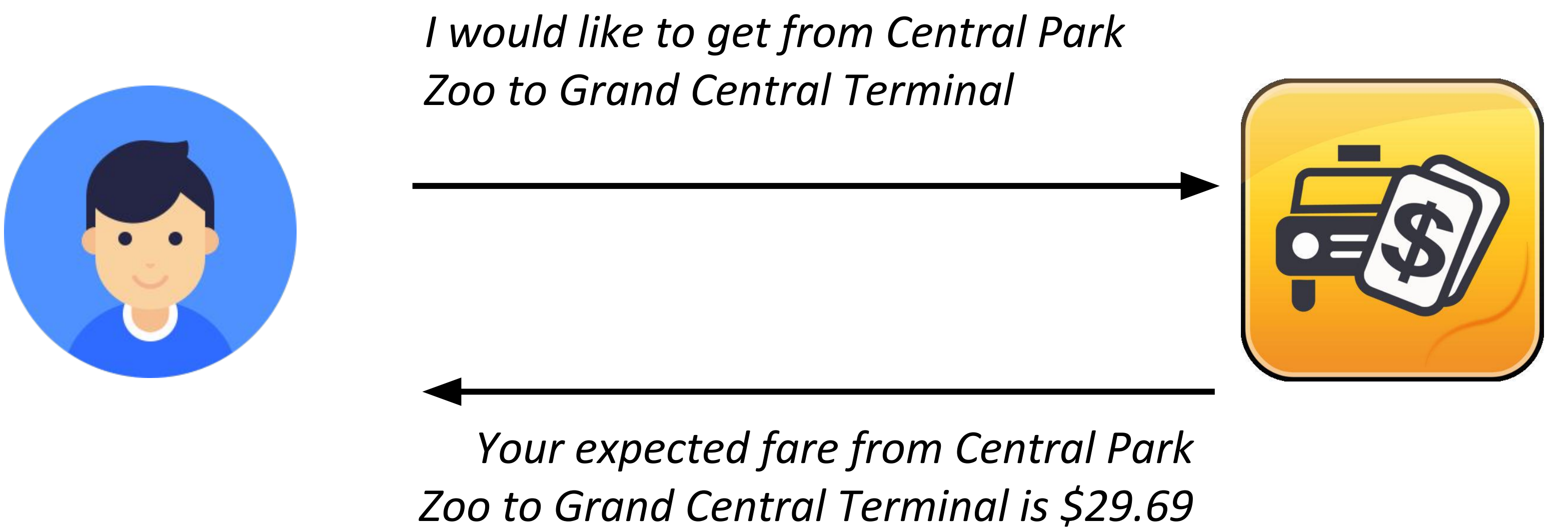
Cloud Machine Learning Engine

ML on Managed Services

- Machine learning training on large datasets are computationally intense
- An increasingly affordable option for users without specialized IT infrastructure is to process ML workloads on the cloud with Managed Services like GCP ML Engine
- 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

Taxi Fare Prediction Application

- Accepts speech queries to get the fare estimate to get from point to point (based on historical data), and returns the result as speech



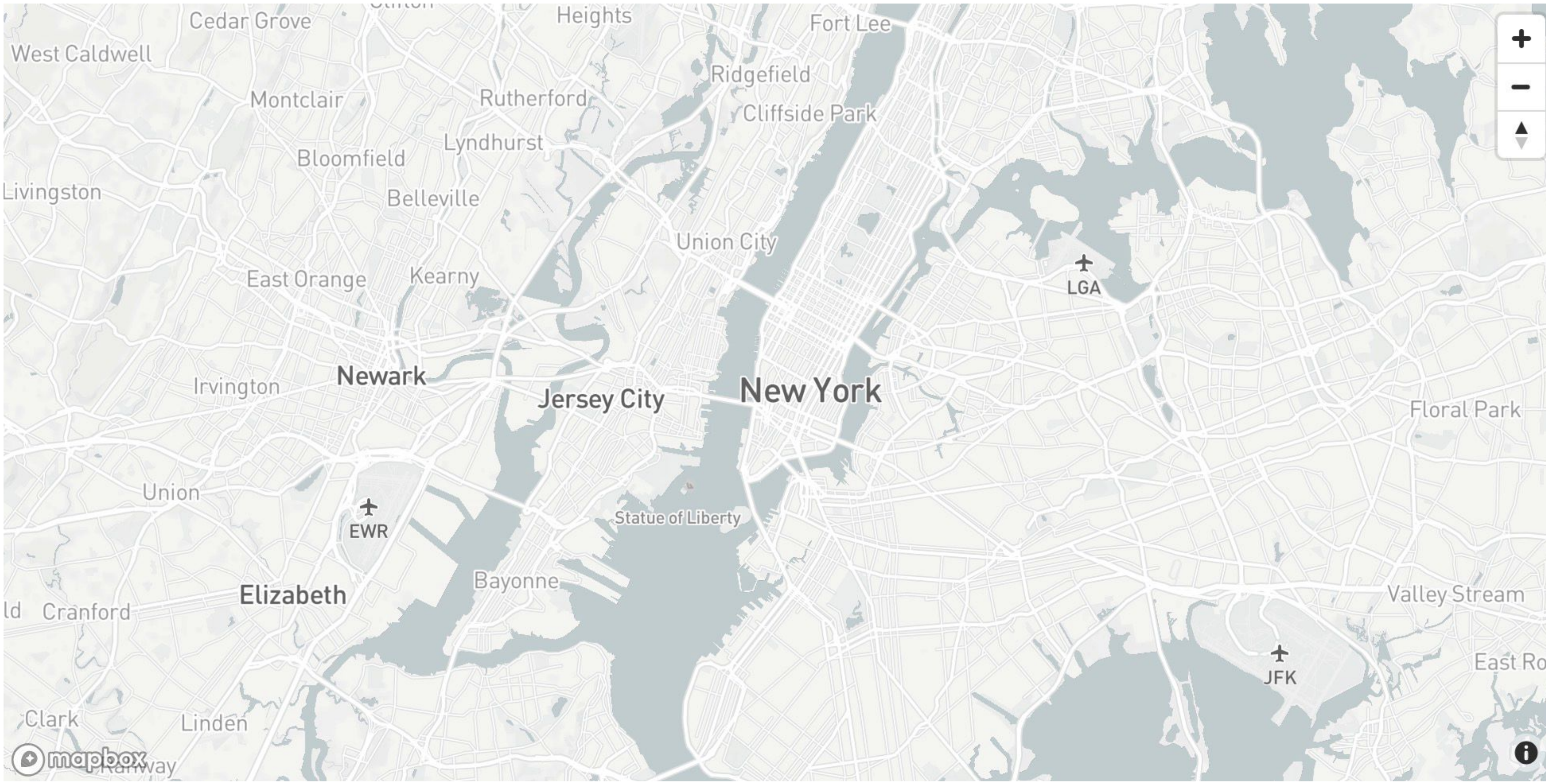
Overview of Tasks

- Task 1: Data Visualization and Feature Engineering
- Task 2: Training, parameter tuning, deploying and serving your model using the Google Cloud ML Engine.
- 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 transfer learning to train a model that accepts custom landmarks as input for prediction

Task 1: Feature Engineering - Data Viz

- 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 - Data Viz



Task 1: Feature Engineering

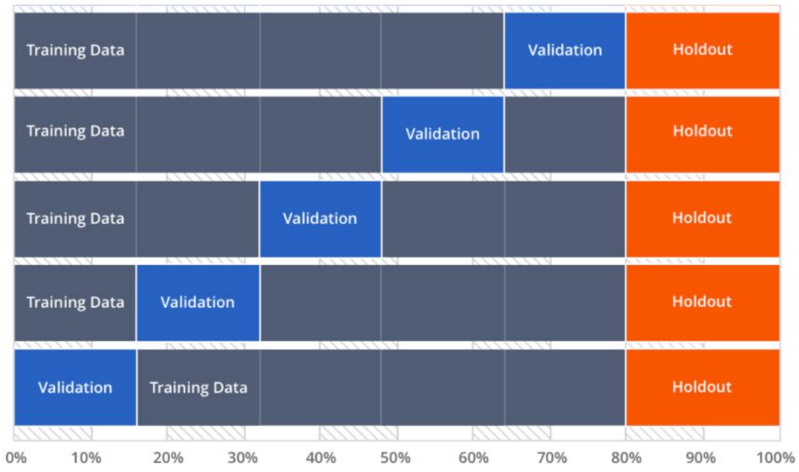
- You are given a small training dataset containing historical data of fare prices in New York City
- Steps to perform
  - Clean data and remove outliers
    - Consider what you learned from the data visualization task
  - Extract or construct meaningful features that will improve performance over the 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 1: Feature Engineering

- Achieve target accuracy, measured by Root Mean Squared Error (RMSE), to earn full credit.

- Grading feedback will tell you how much you need to improve your RMSE to get the next grade.

Task 2: Training, Tuning & Deploying

- Build a complete model with the training dataset, we will leverage ML Engine to perform model training.
- Deploy the trained model to ML Engine.
- Deploy 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 ML Engine
  - Format and return a web response

Task 2: Tuning with GCP ML Engine

- **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 exhaustive Grid Search is expensive

Task 2: Tuning with GCP HyperTune

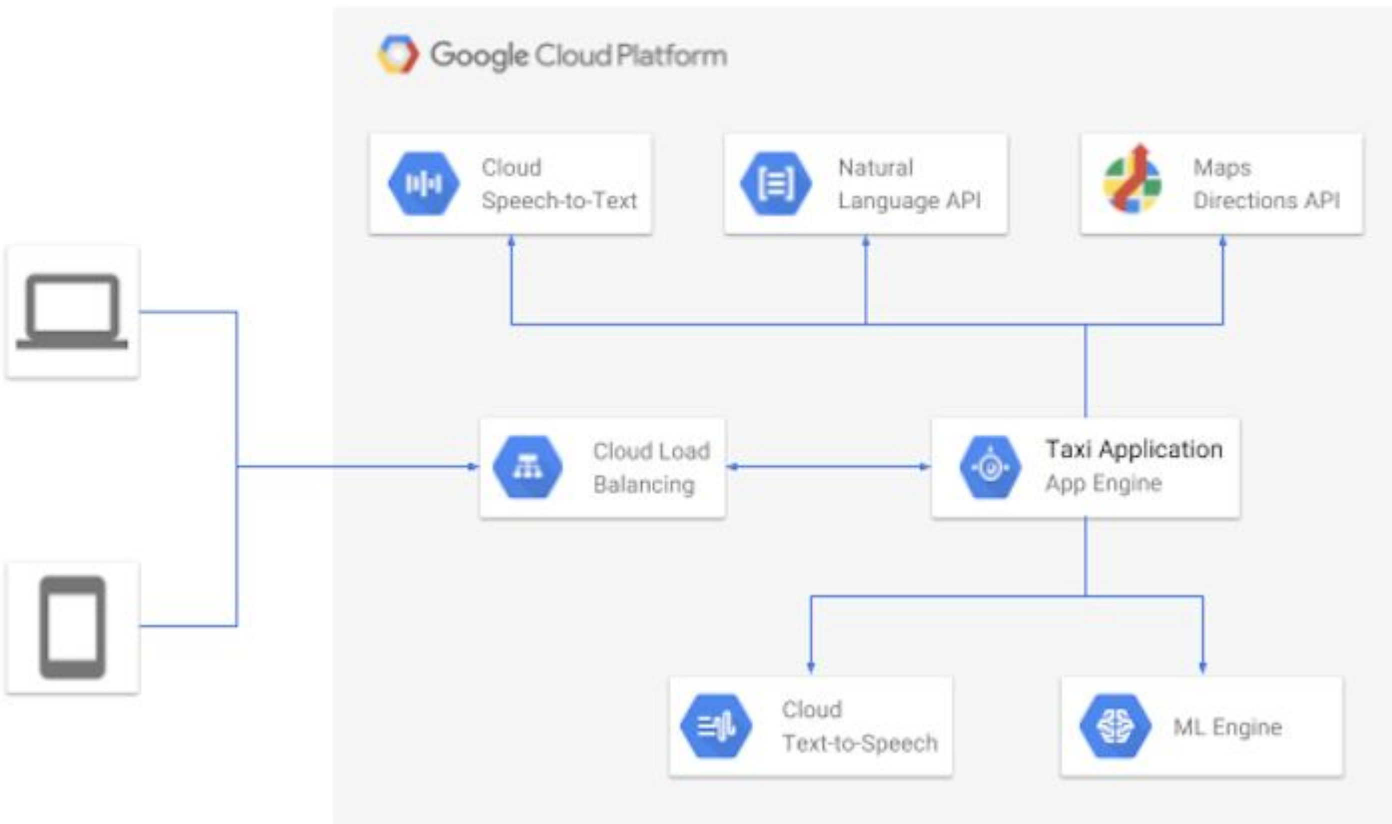
- Black box optimization service (does not need access to the underlying model)
- Need to specify a config yaml file that describes which parameters to tune
- Uses a method called Bayesian Optimization to efficiently search through different combinations of hyperparameters
- An example of a HyperTune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

Task 2: Deploying Models to ML Engine

- **To get a full score in this task, you need to:**
  - (10 points) Complete the following:
    - Enable HyperTune
    - Add at least 3 additional parameters to tune
    - Improve the model performance by at least 3% which is measured by RMSE score.
  - (10 points) Deploy the fare prediction application to Google App Engine (GAE) which can serve web requests correctly.
  - (10 points) 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 rides
    - **/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 ML Engine 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**

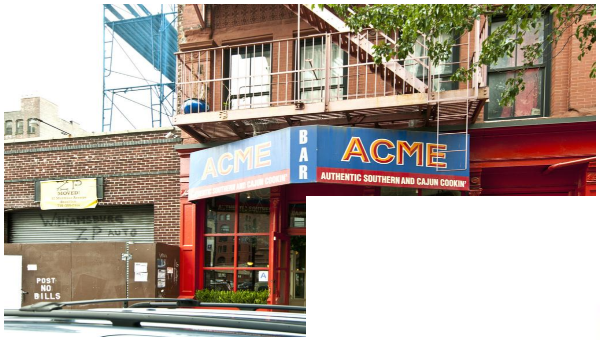
Bonus: Landmark Recognition

...

...

...

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 parameter documentation to understand which parameters can help most.
  - You can change the number of workers for ML Engine to parallelize the training process.
  - Learn to make good estimates for the cost for each run
    - Cost = 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 have passed Task 1, but failed Task 2.
    - You should make sure you have good features first, before trying to play around with filtering outliers.

TEAM PROJECT
Twitter Data Analytics

Team Project Phase 2 Live Test
Top Teams

CaptainMAL	52191.51
GodWeiheng	52052.42
CCNoLife	49864.22
GoGoPowerRanger	48483.37
BESTCC	47626.72

Q1H

BESTCC	61963.9
GoGoPowerRanger	61923.21
CaptainMAL	57639.2
ShotBeforeCC	57510.9
TeamRocket	57314.96

Q1M

Congrats to **CaptainMAL, GoGoPowerRanger** and **BESTCC** for top performance at both HBase and MySQL tests.

Team Project Phase 2 Live Test
Top Teams

CCHunter	21868.6
AutoScalingGroup	21415.72
GoGoPowerRanger	20454.61
nullnullnobug	18779.58
CCaaS	16084.05

Q2H

GoGoPowerRanger	27552.07
CCHunter	23757.92
NtuPuzzleAndDragonLab	21597.64
LongLongName	20245.92
pigeon	19962.12

Q2M

Congrats to **CCHunter** and **GoGoPowerRanger** for top performance at both HBase and MySQL tests.

Team Project Phase 2 Live Test
Top Teams

GodWeiheng	5357.58
nullnullnobug	5069.03
NtuPuzzleAndDragonLab	4736.36
CCHunter	4592.7
obobobobobobob	4375.31

Q3H

LongLongName	8913.7
Could Compute	7225.1
SimpleNaive	7107.97
R3-D3	7077.07
NtuPuzzleAndDragonLab	6736.8

Q3M

Congrats to **NtuPuzzleAndDragonLab** for top performance at both HBase and MySQL tests.

Team Project Phase 2 Live Test
Top Teams

Congrats to:

LongLongName,
NtuPuzzleAndDragonLab,
nullnullnobug and
CCaaS

for achieving a full score for the live test!

Team Project - Phase 3

- Use only **AWS managed services** for all queries.
- Development budget: $100
  - Penalty for lavishness: >$150
- Live test:
  - Per-hour-budget: $1.28 (included in $100)
- Perform ETL on your beloved GCP and Azure

Cloud Managed Services

- Managed services remove the burden from having to operate the provisioned cloud infrastructure.

- Management of the tools such as monitoring, patching, security, backup are offered as part of the service.

Team Project - Phase 3

- RPS targets have been changed →
  - Q1: 30000
  - Q2: 12000
  - Q3: 5000
- Teams should NOT use any EC2 VMs or EBS volumes.
- Rule of thumb:
  - If you see anything in EC2 dashboard, stop.
  - If you are doing `sudo apt install mysql-server`, stop.
- Teams should explore the managed services provided by AWS to come up with a solution.
- Teams are required to use Terraform (unless Terraform does not have support for your particular managed service)

Team Project General Hints

- No EC2 VMs and EBS volumes in the live test!
  - Nonetheless, you can use those to do verification or comparison to the hosted service you built before in the development process.

- You can check the EC2 web console after launching the managed service to verify if the managed service is allowed
  - Example 1: Lambda is allowed since it there will be no EC2 instances visible in the web console while using.
  - Example 2: EMR is not allowed because there are master and slave machines in the web console.

Team Project General Hints

- One option would be to split the services into web-tier and storage-tier and choose different managed services.
  - If so, the compatibility of these two services should be taken into account.

- Consider the different characteristics of queries to decide what kind of managed services to use.

- High performance/cost ratio is valued.
  - Try your best to achieve the highest possible ratio.

Team Project Time Table

Phase (and query due)	Start	Deadlines	Code and Report Due
Phase 1 • Q1, Q2	Monday 02/25/2019 00:00:00 ET	**Checkpoint 1, Report: Sunday 03/11/2019 23:59:59 ET** **Checkpoint 2, Q1: Sunday 03/25/2019 23:59:59 ET** **Phase 1, Q2: Sunday 03/31/2019 23:59:59 ET**	Phase 1: Tuesday 04/02/2019 23:59:59 ET
Phase 2 • Q1, Q2,Q3	Monday 04/01/2019 00:00:00 ET	**Sunday 04/14/2019 15:59:59 ET**	
Phase 2 Live Test (Hbase **AND** MySQL) • Q1, Q2, Q3	**Sunday 04/14/2019 17:00:00 ET**	Sunday 04/14/2019 23:59:59 ET	**Tuesday 04/16/2019 23:59:59 ET**
Phase 3 • Q1, Q2, Q3 (Managed services)	Monday 04/16/2019 00:00:00 ET	Sunday 04/28/2019 15:59:59 ET	
Phase 3 Live Test • Q1, Q2, Q3 (Managed services)	Sunday 04/28/2019 17:00:00 ET	Sunday 04/28/2019 23:59:59 ET	Tuesday 04/30/2019 23:59:59 ET

Upcoming Deadlines

- Project 4.2: Machine Learning on the Cloud
  - Due Sunday, April 21, 2019, 11:59 PM ET
- Team Project : Phase 3
  - Live-test at: Sunday April 28, 2019 3:59 PM ET
  - Code and report due: Tuesday April 30, 2019 11:59 PM ET

Questions?