CMPE 258 Haley Feng
Professor Vijay Eranti

HW #2.2 AutoML Vision and Timeseries

➢ Vision

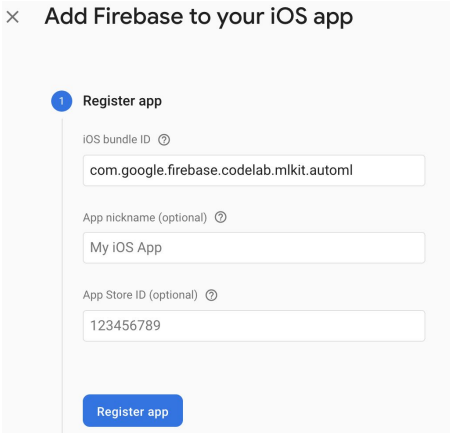
(https://codelabs.developers.google.com/codelabs/automl-vision-edge-in-mlkit#0)

I. Setup

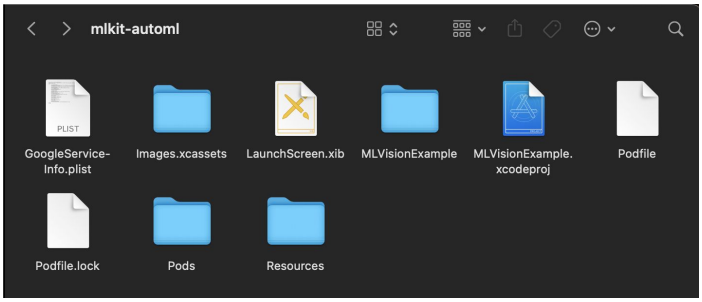
Packages download:

- Zip file with source code, the training dataset
- Google-services.json config file
- XCode from App Store
- Cocoapods

On the Firebase console, create a project name ML Kit Codelab and add firebase to iOS app with bundle ID `com.google.firebase.codelab.mlkit.automl`



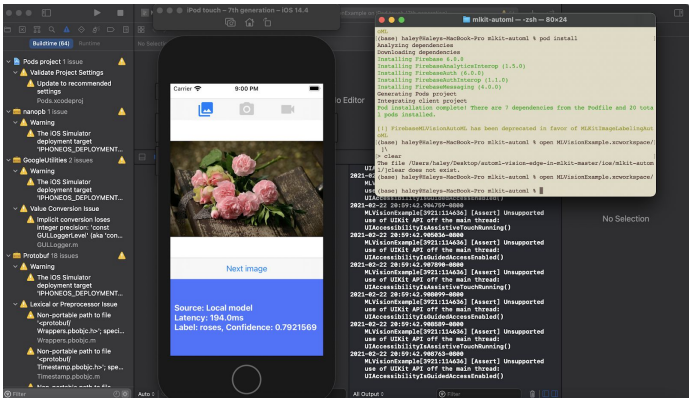
Move Google-services.json file into project folder in XCode



II. Running iOS App

Using the pre-trained model inside the folder Resources/automl/ (skipping the steps of creating flower dataset and training the model), open the terminal and type in the following:

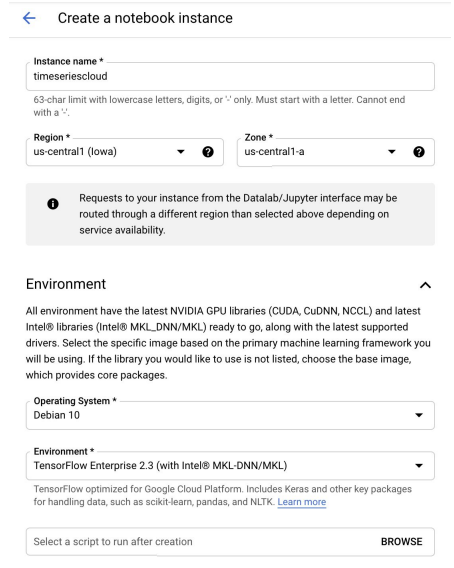
```
cd /Users/haley/Desktop/automl-vision-edge-in-mlkit-master/ios/mlkit-automl

pod install
open MLVisionExample.xcworkspace/
```

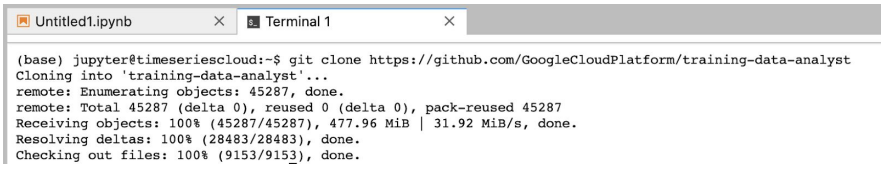
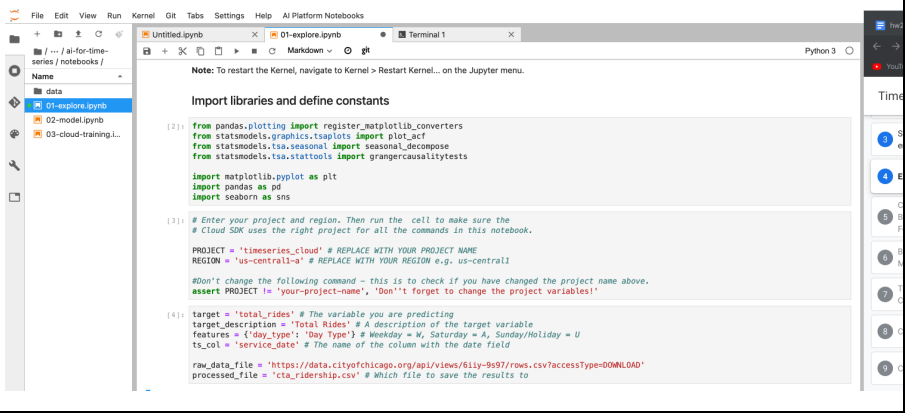
	Podfile source 'https://github.com/CocoaPods/Specs.git' platform :ios, '9.0' use_frameworks! target 'MLVisionExample' do pod 'FirebaseMLVision', '0.16.0' pod 'FirebaseMLCommon', '0.16.0' pod 'FirebaseMLVisionAutoML', '0.16.0' pod 'Firebase/Analytics' pod 'Firebase/Core' pod 'Firebase/Messaging' pod 'Firebase/Auth' end
After the pod finish installing the required modules inside the Podfile, run `open MLVisionExample.xcworkspace/` in the terminal to open the XCode workspace and click on the upper left play button to generate a simulation of the app	

➢ Timeseries Forecast

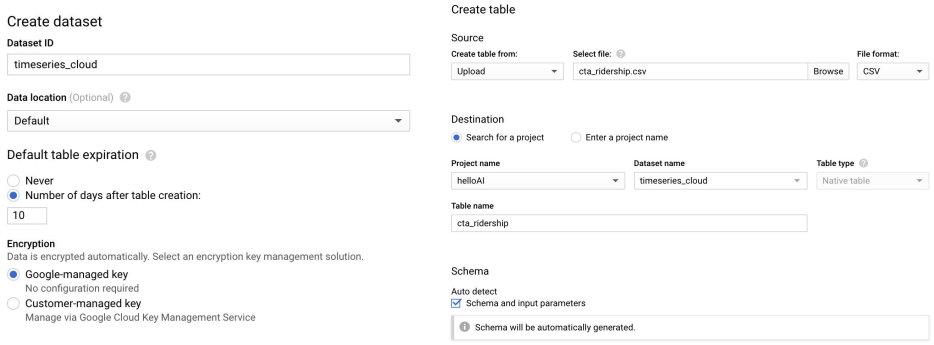
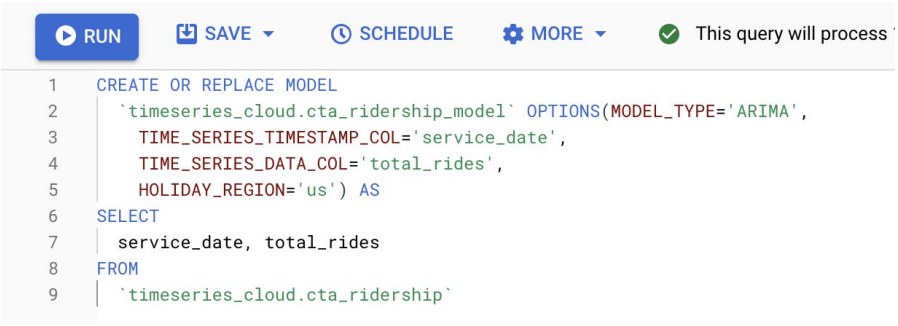
(https://codelabs.developers.google.com/codelabs/time-series-forecasting-with-cloud-ai-platform#0)

I. Setup

Create an instance on the AI Platform Notebooks section with the following setup: • Instance name = timeseriescloud • Zone = us-central1-a • Environment = TensorFlow:2.3 • GPUs = None	

	
Clone source material using the terminal and open the provided 01-explore notebook	
Download dataset and run the 01-explore notebook with data evaluation/exploration	

II. Model in BigQueryML

Create a dataset name *timeseries_cloud* and create a table name *cta_ridership.csv* with the dataset uploaded	
Create an ARIMA time series model in the BigQuery editor console using the dataset feature of service_date and total_rides	

The query ML.EVALUATE will return candidate models sorted by AIC score (lowest to highest). The first model is the best model with a 1 p (order of auto-regression), 1 q (order of integration), 4 q (order of moving average)

```
SELECT
  *
FROM
  ML.EVALUATE(MODEL `timeseries_cloud.cta_ridership_model`)
```

Query results SAVE RESULTS EXPLORE DATA

Query complete (0.3 sec elapsed, 0 B processed)

Job information | Results | JSON | Execution details

Row	non_seasonal_p	non_seasonal_d	non_seasonal_q	has_drift	log_likelihood	AIC	variance	seasonal_periods
1	1	1	4	true	-84343.91298029698	168701.82596059397	2.1214766324672794E9	WEEKLY
								YEARLY
2	1	1	4	false	-84345.76278035615	168703.5255607123	2.1226282591786644E9	WEEKLY
								YEARLY
3	4	1	1	true	-84346.86918283005	168707.7383656601	2.1232853081307085E9	WEEKLY
								YEARLY

Use the ML.FORECAST query to forecast the total rides for 7 days using the model

```
SELECT
  *
FROM
  ML.FORECAST(MODEL `timeseries_cloud.cta_ridership_model`, STRUCT(7 AS horizon))
```

Query results SAVE RESULTS EXPLORE DATA

Query complete (0.3 sec elapsed, 23.4 KB processed)

Job information | Results | JSON | Execution details

Row	forecast_timestamp	forecast_value	standard_error	confidence_level	prediction_interval_lower_bound	prediction_interval_upper_bound	confidence_interval_lower_bound
1	2020-01-01 00:00:00 UTC	662436.4424369269	46059.49014554253	0.95	572322.980240453	752549.9046334007	572322.980240453
2	2020-01-02 00:00:00 UTC	1029641.4669424891	46276.328347693256	0.95	939103.76989082	1120179.1639941582	939103.76989082
3	2020-01-03 00:00:00 UTC	1201660.2034356925	47233.43871922012	0.95	1109249.9600529654	1294070.4468184195	1109249.9600529654
4	2020-01-04 00:00:00 UTC	651095.9776391207	48157.99332862347	0.95	556876.8819095747	745315.0733686666	556876.8819095747
5	2020-01-05 00:00:00 UTC	467394.91846646497	48621.50963880497	0.95	372268.97250121285	562520.8644317171	372268.97250121285
6	2020-01-06 00:00:00 UTC	1158999.319539823	48869.23710364581	0.95	1063388.705171438	1254609.9339082083	1063388.705171438
7	2020-01-07 00:00:00 UTC	1127789.5651062205	49011.66149084522	0.95	1031900.3033930386	1223678.8268194026	1031900.3033930386

III. Custom Forecasting Model

Open the 02-model notebook from source material with the exercise for:

- Removing outliers from time series data
- Long Short Term Memory (LSTM)
- Convolutional Neural Network (CNN)
- Random Walk - ARIMA model
- Season Naïve -SARIMA
- Exponential Smoothing method

ML Models

In this section, you will build models using popular neural network architectures for time-series data.

Long Short Term Memory (LSTM)

```
[20]: # Reshape test data to match model inputs and outputs

X_train = X_train_reframed.values.reshape(-1, n_input_steps, n_features)
X_test = X_test_reframed.values.reshape(-1, n_input_steps, n_features)
y_train = y_train_reframed.values.reshape(-1, n_output_steps, 1)
y_test = y_test_reframed.values.reshape(-1, n_output_steps, 1)
```

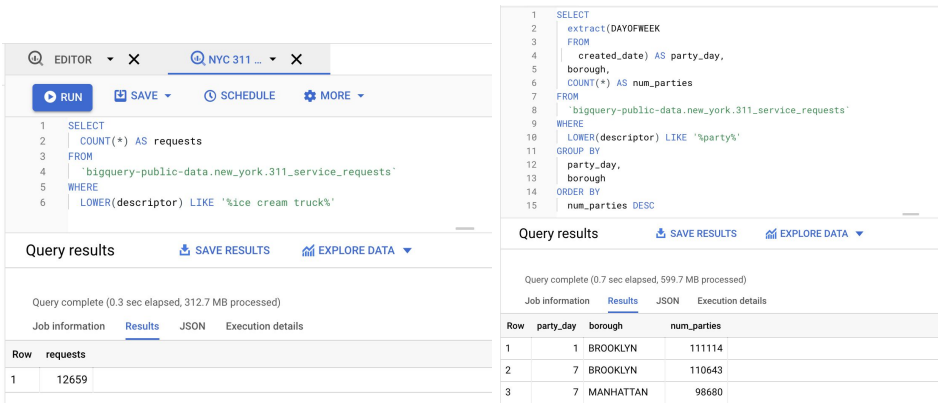
TODO 2: Update the LSTM architecture

Try increasing and decreasing the number of LSTM units and see if you notice any accuracy improvements.

You can use hyper-parameter tuning to search for optimal values, but that's outside the scope of this lab.

```
[21]: # Try increasing and decreasing the number of LSTM units and see if you notice any accuracy imp
# Run the next cell to evaluate the results in more detail.

model = Sequential([
    LSTM(64, input_shape=[n_input_steps, n_features]),
    Dense(n_output_steps)])
```

IV. Challenge: 311 service requests

Get access to the City of NY 311 Service Requests dataset and run queries for it	
Duplicate the previous notebooks and replace the new dataset in it	[12]: ```sql = """ SELECT COUNT(unique_key) as y, DATE_TRUNC(DATE(created_date), month) as ds FROM `bigquery-public-data.new_york_311.311_service_requests` GROUP by ds ORDER BY ds asc """ df_inc = client.query(sql).to_dataframe() df_inc.head()``` [12]: y ds 0 182117 2010-01-01 1 159489 2010-02-01 2 198639 2010-03-01 3 162854 2010-04-01 4 158039 2010-05-01 [13]: ```target = 'y' # The variable you are predicting target_description = 'Count of Incidents' # A description of the target variable #features = {'day_type': 'Day Type'} # Weekday = W, Saturday = A, Sunday/Holiday = U ts_col = 'ds' # The name of the column with the date field #raw_data_file = 'https://data.cityofchicago.org/api/views/6iiy-9s97/rows.csv?accessType=DOWNLOAD' #processed_file = 'cta_ridership.csv' # Which file to save the results to```