

Google AI Platform Pipelines

A Cook-Book for Healthcare & Life Science



Google Cloud Platform

Predicting Peptide for Vaccine Candidate (Covid-19)

[Presentation](#) and [Recording](#)

Motivated by Proposal for [kaggle](#) COVID challenge and [Google Project Uplift](#)

Contributors:

[Vadim Astakhov](#)

[Jignesh Mehta](#)

Last Updated: April 15, 2020

[Presentation](#) and [Recording](#)

Table of Contents

[Introduction](#)

[Proposal](#)

[Technical Assets](#)

[General Problem](#)

[Immune Response for Viral Infection](#)

[Key Challenges](#)

[Which HLA we might want to consider](#)

[What Peptides should we consider](#)

[ML Modeling for Peptide Prediction](#)

[Dataset \(+BQ Public Dataset\)](#)

[Some additional COVID-19 genomics and proteomics data:](#)

[Building a Model with AutoML Table](#)

[Model Exploration with TensorBoard](#)

[Build ML with BigQuery](#)

[Accelerate with GCP AI Pipeline](#)

[Architecture](#)

[AI Pipeline Template](#)

[Deploying Covid19 Pipelines Template](#)

[Deploy Model to Serve Prediction](#)

[Additional Examples](#)

[Future Development](#)

[References](#)

[A: Domain References](#)

[B: Technical References](#)

Sign-off grid

Ldap	Role	Date

Changelog

Editor	Comments	Date
..		
jigmehta	Proposal, Binding details, pipeline architecture	03/27/2020
astakhov	Models to explore HLA-A*02:01 COVID peptides binding	03/28/2020
jigmehta	Dataset, BQML model, AI Pipeline details	03/30/2020
astakhov	Add references for data sets, and publications	03/30/2020
jigmehta	Pipeline deployment and additional examples	04/07/2020

Links

1. Introduction

Coronavirus disease 2019 (COVID-19) is a kind of viral pneumonia with an unusual outbreak in Wuhan, China, in December 2019, which is caused by severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2). With a reproduction factor (R_0 - R naught) of 2.2, Covid-19 has already spread across boundaries and infected more than half a million people in the world at the time of this writing.

COVID-19 is not the first severe respiratory disease outbreak caused by the coronavirus. According to the World Health Organization (WHO), viral diseases continue to emerge and represent a serious issue to public health. In the last twenty years, several viral epidemics such as the severe acute respiratory syndrome coronavirus (SARS-CoV) in 2002 to 2003, and H1N1 influenza in 2009, have been recorded. Most recently, the Middle East respiratory syndrome coronavirus (MERS-CoV) was first identified in Saudi Arabia in 2012.

Intense research on this family of viruses has been conducted. In this [paper](#) published by Wuhan BSL-4 lab scientists in 2013 claimed that they can grow viruses in human cells (China [BSL4 lab](#) located in Wuhan). Another work published in the prominent scientific journal Nature Medicine demonstrated the potential for bat Coronavirus to [emerge on humans](#) by creation of a chimeric SARS-like virus. This paper has scientists discussing the risks of gain-of-function research and [triggered debate](#) among the general public. The recent [publication](#) in Nature Medicine argues that if SARS-CoV-2 pre-adapted in another animal species, then there is the risk of future re-emergence events. All of that emerging information prompts us to start thinking what we can proactively do to help in the prevention of future zoonotic events.

Proposal

We are proposing a [GCP AI Pipeline based framework](#) to help research and investigate potential vaccine candidates. The AI Pipeline templates for HLA-Peptide binding helps predict possible antigen for vaccine testing. AI pipeline addresses top two challenges for researchers:

1. Help accelerate AI/ML model testing and deployment by plug and play of dataset and ML models.
2. Scale AI platform to large scale dataset to identify and sort list antigen candidates for vaccine test.

With this approach we are not just building a system to research for the Covid-19 but also for the future to discover novel insights and potential therapeutic targets for combating the SARS-CoV-2 and other viral infections.

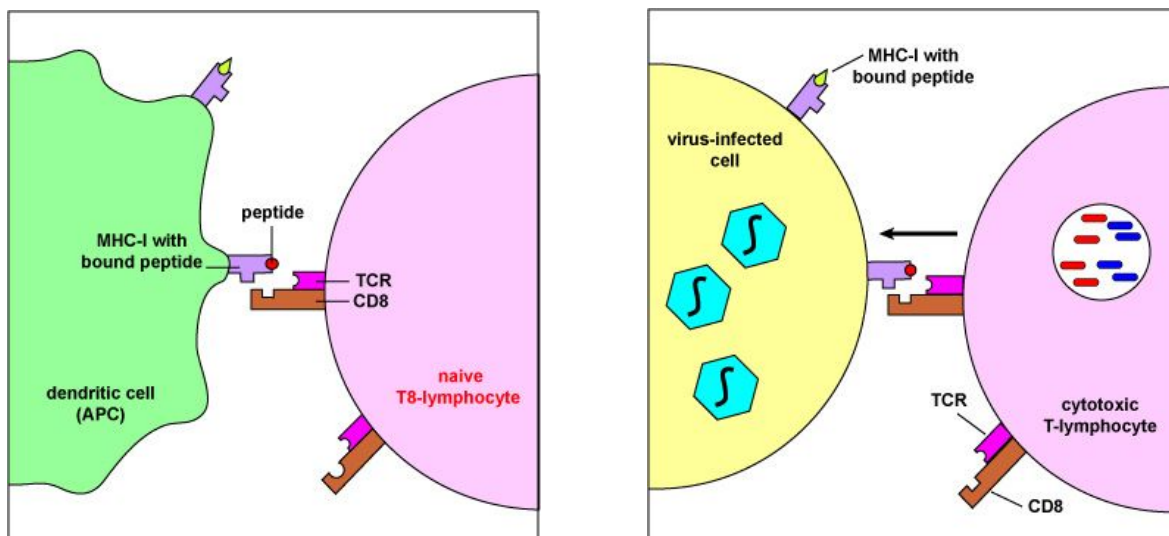
Technical Assets

- a. Peptidic Epitope data set for BQ Public data set repository
 - i. Subset of data → coronavirus → BQ Dataset
- b. BQ ML Model notebook and AI Pipeline notebook → Terra notebook examples
- c. AI Platform ML Pipeline template for bioscience research / biotech customers

2. General Problem

Immune Response for Viral Infection

At a high-level, when the virus enters the cells, some of its protein fragments (**peptides**) called antigen will be presented to the immune system. These antigenic peptides first need to be able to bind with major histocompatibility complexes **MHC** molecule (which is called human leukocyte antigen (**HLA**) in humans) and then recognized by virus-specific cytotoxic T lymphocytes cells (**T-Cells**). Thus the immune system can recognise and destroy virus infected cells. Chart below shows the MHC-Peptide binding presentation to T-Cell, T-Cell receptors (**TCR**) needs to be able to accept binding to produce an immune reaction that kills infected cells.



Source: <https://bio.libretexts.org>

Basically, the immune system recognises some proteins which compose the virus, where the recognition process is facilitated by the process of binding virus peptide to the HLA in humans. Such binding makes the virus “visible” to the immune system.

Key Challenges

The HLA-Peptide binding is usually characterized by very high selectivity achieved through the interaction of the HLA with several critical (anchoring) residues of a peptide. Thus, despite the fact that biodegradation of antigenic proteins can theoretically produce a very large diversity of peptides, the actual number of them selectively bound to a specific HLA allele is very limited.

This makes it non-trivial and a very important goal to identify those specific fragments of protein sequences that are capable of selective interaction with specific HLA alleles. It is believed that the ability of predicting HLA binding is also an essential step of ‘in silico’ vaccine development.



If we identify possible virus peptides which bind HLA then we can use them as potential candidates for peptide vaccines to train the human immune system to battle the virus.

Problem is that there are often hundreds of virus peptides which do not bind to HLA. Classification of those peptides by binding score and binding quality would speed up the selection potential candidates for vaccine development. Identifying possible antigen candidates through AI drastically reduces the number of bindings to test for a viable vaccine solution.

We are proposing to leverag [Google AI Pipeline platform](#) to build generic pipelines to generate candidates for vaccine development.

[Which HLA we might want to consider](#)

Previous research shows HLA-DR0301, HLA-Cw1502 and HLA-A*0201 alleles are related to the protection from SARS infection. These researches might be valuable clues for the prevention, treatment, and mechanism of COVID-19. For AI/ML models, one can slide learning/prediction by HLA Allele or consider it as a categorical feature to include in learning.

Reference to look for highly possible HLA Alleles:

1. Predicting [HLA-A2 binding peptides](#)
2. [HLA Class consideration](#) for coronavirus

[What Peptides should we consider](#)

For correct immune response to viral infected cells, it is important to understand the binding affinity between peptide and MHC Allele. Coronavirus S protein has been reported as a significant determinant of virus entry into host cells. The envelope spike glycoprotein binds to its cellular receptor, ACE2 SARS-CoV-2. Thus we might want to consider binding affinity of S protein peptides. Research has shown that for a set of Allele most likely to bind with 9-mer or 10-mer peptides. For our example, we will filter for 9-mer and 10-met peptides for ML models.

Reference to look for HLA-Peptide binding affinity:

1. New:Cov-19 - [Candidate Target for immune system](#)
2. [Peptide - HLA Binding CNN model](#)

3. ML Modeling for Peptide Prediction

Dataset (+BQ Public Dataset)

There are multiple data sets available through various educational institutions for research on HLA-Peptide bindings (some of them can be found [here](#)). For our examples, we will be using a data set from Immune Epitope Database and Analysis Resource (<http://www.iedb.org/>). You can bring a subset of data of your interest from IEDB into BigQuery for analysis. We have already created a data set to accelerate your research.

BQ Public Dataset

- **covid19_iedb_full_viral** (Dataset with epitopes across multiple viral and antigen)
 - antigen_full (table with antigen reference)
 - epitope_full (table with HLA-Peptide binding information with binding score)
 - mhc_lignad_full (table with linear peptide details with source/host organism)
 - amino_acid (table with amino acid properties - cross features for modeling)
- **covid19_iedb_corona_subset** (Dataset subset for coronavirus samples)
 - antigen (table with antigen reference)
 - epitope (table with HLA-Peptide binding information with binding score)
 - mhc_lignad (table with linear peptide details with source/host organism)
 - Mhc_qual_feature (table with sample featured columns for amino acid position)
- **covid19_iedb_influenza_subset** (Dataset subset for influenza virus samples)
 - antigen (table with antigen reference)
 - epitope (table with HLA-Peptide binding information with binding score)
 - mhc_lignad (table with linear peptide details with source/host organism)

We will be using **covid19_iedb_corona_subset** data for ML modeling examples and for AI Pipeline to demonstrate the process.

Some additional COVID-19 genomics and proteomics data:

- [Severe acute respiratory syndrome coronavirus 2 isolate Wuhan-Hu-1, complete genome](#)
- [SWISS-MODEL | Severe acute respiratory syndrome coronavirus 2](#) - a fully automated protein structure homology-modelling server which is modelled the full SARS-CoV-2 proteome based on the NCBI reference sequence **NC_045512** which is identical to GenBank entry **MN908947**.
- [COVID-19 proteins](#)
- [Structure of 2019-nCoV spike protein](#)
- [Protein Data Bank for COVID-19](#)
- [Predict protein- structure \(model and data\)](#)
- [Global Initiative on Sharing All Influenza Data](#)

Building a Model with AutoML Table

AutoML can be used as a quick way to explore data and try to understand what models can be used to perform inference.

One of the advanced features of AutoML Table is to automatically perform common feature engineering tasks for you and produce state-of-the-art models ([see features for more details](#)) with one click.

Key Advantage of AutoML:

- Takes your dataset and starts training for multiple model architectures at the same time
- Determine the best model architecture for your data
 - Linear
 - Feedforward deep neural network
 - Gradient Boosted Decision Tree
 - AdaNet
 - Ensembles of various model architectures
- Export models as docker image.
- Reverse engineering and model exploration in [TensorBoard](#).

As a test example, we will leverage the “mhc_qual_feature” table which we created to have HLA alleles and peptides with known binding affinity.

Following query shows a featured table which will be used for AutoML Table.

This table is

generated based on other base table with filter of only those linear peptides with

length of 9 or 10 mers. The result shows 10 position for indexing amino acids in a

given peptide as well as binding measures with quality and score.

```
select * from `corona.mhc_qual_feature`
```

Row	ligand_id	peptide	result_type	result	Allele	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	result_score	result_quality
1	1717014	ILLNKHID	Positive-Low	0	HLA-A*02:01	0	0	0	0	0	0	0	0	0	0	4000.0	1
2	1025027	AACAMLLVK	Negative	0	HLA-A*68:01	1	1	5	1	13	11	11	20	12	0	16800.0	0
3	1025025	AACAMLLVK	Negative	0	HLA-A*31:01	1	1	5	1	13	11	11	20	12	0	10600.0	0
4	1025026	AACAMLLVK	Negative	0	HLA-A*33:01	1	1	5	1	13	11	11	20	12	0	77700.0	0

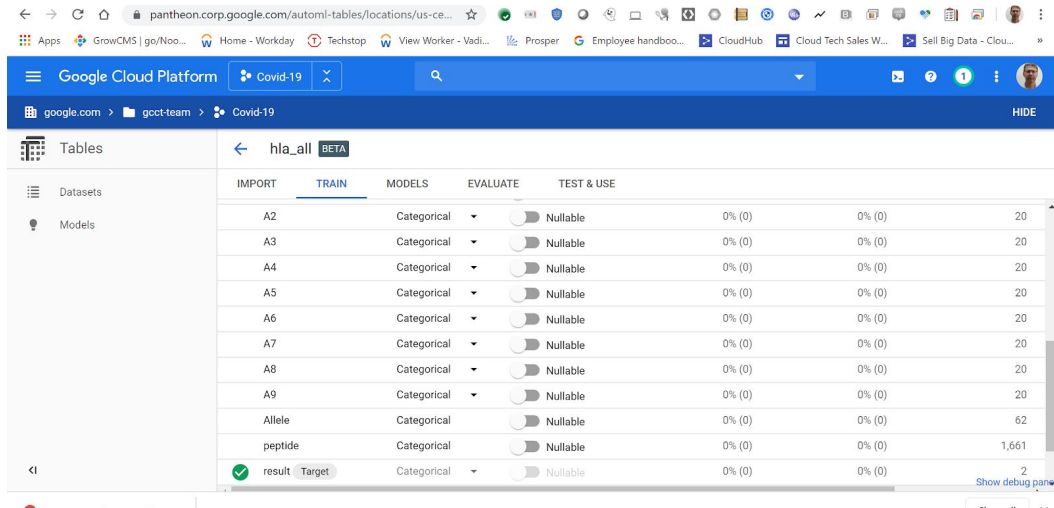
We will encode amino_acid position in a given peptide as a categorical feature (1 for A, 2 for B, 3 for Ca and so on) and we will explore if a specific sequence of amino acids in a peptide has good binding properties. There are many other features we can generate and use, such as amino acid polarity, weight and AutoML is a great way to quickly explore those features across multiple models which will be trained in parallel by AutoML.

Data set consists of 63 MHC alleles, 1687 viral epitopes, and 12698 MHC ligands.

Data set consists of 63 MHC alleles, 1687 viral epitopes, and 12698 MHC ligands. We can also filter data set down to HLA-A*02:01 which has been previously [identified](#) as alleles related to the protection from SARS infection.

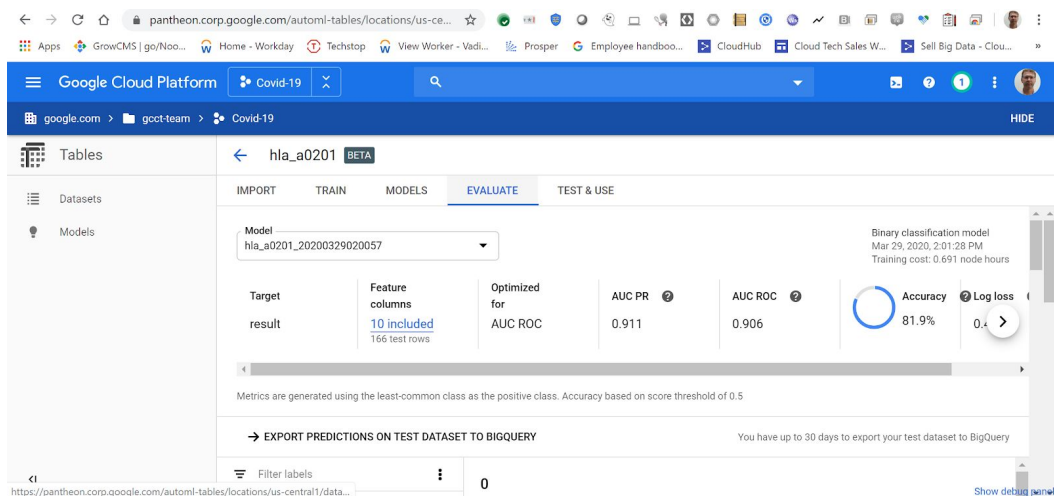
*(SELECT mhc_ligand_id, epitope_id, description, Name, Qualitative_Measure, Quantitative_measurement, Allele_Name FROM `covid-19-271622.corona.mhc_ligand` where Allele_Name = "HLA-A*02:01")*

Then, we will define the “result” column as a categorical target to predict 1-binding and 0- no binding respectively and will quickly get preliminary results which can be a starting point for model enhancement.



The screenshot shows the Google Cloud Platform console for the 'hla_all' table. The 'TRAIN' tab is selected, showing a list of columns and their properties. The 'result' column is highlighted as the target.

Column	Type	Nullable	0%	1%	Count
A2	Categorical	Nullable	0%	0%	20
A3	Categorical	Nullable	0%	0%	20
A4	Categorical	Nullable	0%	0%	20
A5	Categorical	Nullable	0%	0%	20
A6	Categorical	Nullable	0%	0%	20
A7	Categorical	Nullable	0%	0%	20
A8	Categorical	Nullable	0%	0%	20
A9	Categorical	Nullable	0%	0%	20
Allele	Categorical	Nullable	0%	0%	62
peptide	Categorical	Nullable	0%	0%	1,661
result	Categorical	Nullable	0%	0%	2



The screenshot shows the Google Cloud Platform console for the 'hla_a0201' table. The 'EVALUATE' tab is selected, showing model performance metrics.

Metric	Value
Model	hla_a0201_20200329020057
Target	result
Feature columns	10 included 166 test rows
Optimized for	AUC ROC
AUC PR	0.911
AUC ROC	0.906
Accuracy	81.9%
Log loss	0.4

Binary classification model
Mar 29, 2020, 2:01:28 PM
Training cost: 0.691 node hours

Metrics are generated using the least-common class as the positive class. Accuracy based on score threshold of 0.5

→ EXPORT PREDICTIONS ON TEST DATASET TO BIGQUERY

You have up to 30 days to export your test dataset to BigQuery

Once the model is trained, we can export the result. You'll find the export option under **TEST & USE**. (See the [documentation](#) for details on the export process).



Model Exploration with TensorBoard

We will be leveraging TensorBoard for AutoML model reverse engineering and exploration. This stem might provide initial insight of the model complexity and provide Bioinformaticians with leads on model enhancement.

You can view your exported custom model in [TensorBoard](#). This requires a conversion step. You will need to have TensorFlow 1.14 or 1.15 installed to run the conversion script.

Then, download [this script](#), e.g. via

```
curl -O  
https://raw.githubusercontent.com/amygdala/code-snippets/master/ml/automl/tables/model_  
export/convert_oss.py,
```

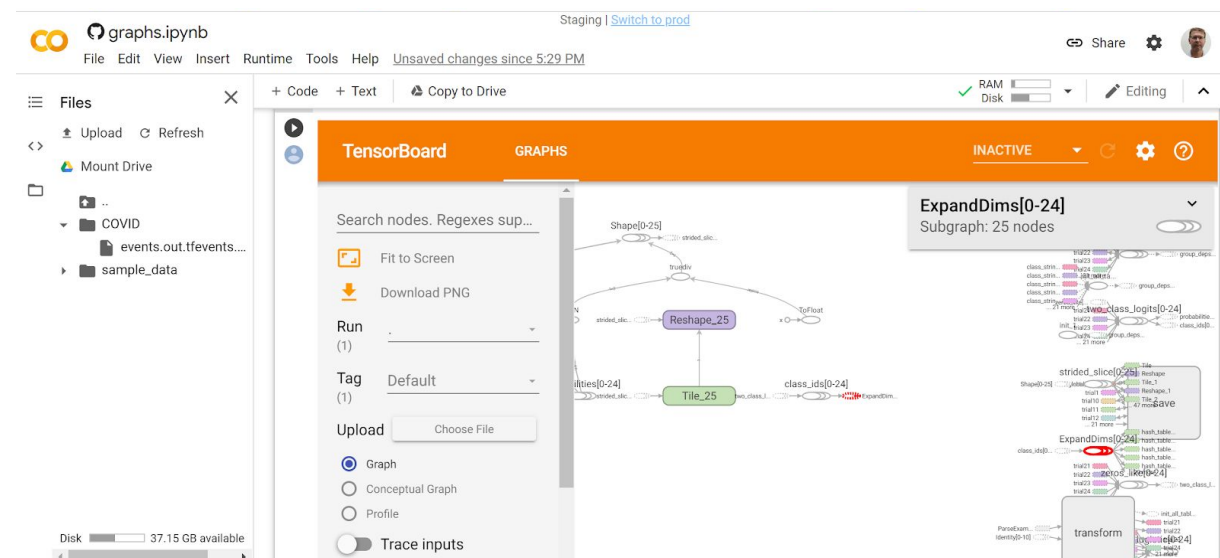
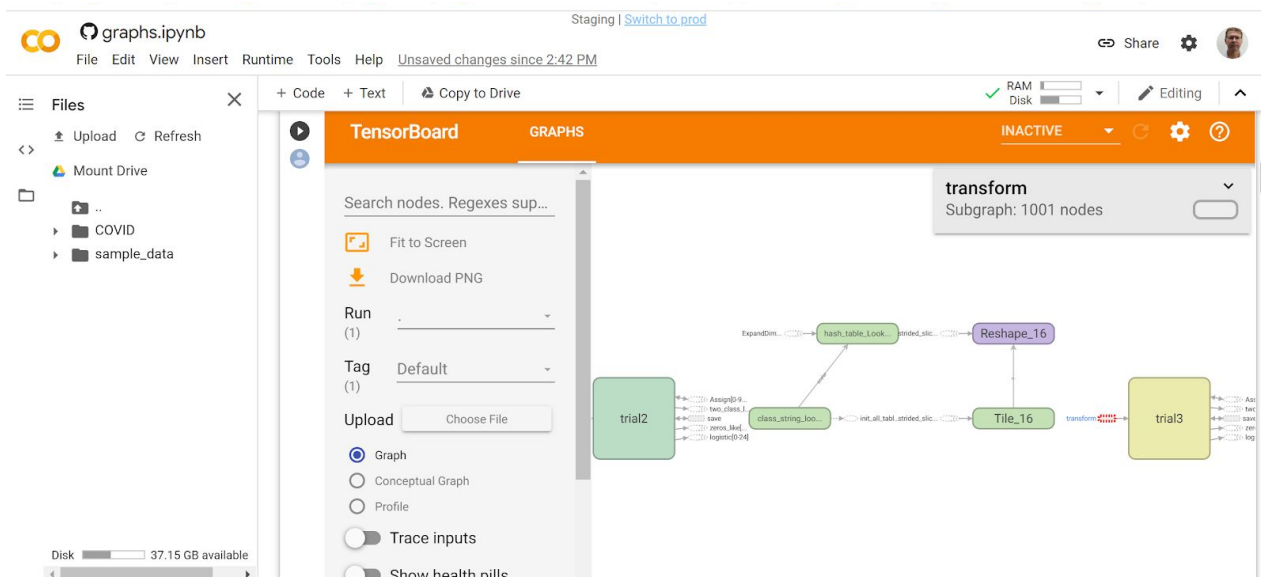
to the parent directory of `model_export`. Create a directory for the output (e.g. `converted_export`), then run the script as follows:

```
mkdir converted_export  
python ./convert_oss.py --saved_model  
./model-export/tbl/<your_renamed_directory>/saved_model.pb --output_dir converted_export
```

Then, point TensorBoard to the converted model graph:

```
# Load the TensorBoard notebook extension  
  
%load_ext tensorboard  
  
tensorboard --logdir=converted_export
```

You will see a rendering of the model graph, and can pan and zoom to view model sub-graphs in more detail.



Exploring automatically generated models can give data scientists and data analysts new leads on how models can be enhanced.

Build ML with BigQuery

One of the advanced features of BigQuery is to build machine learning models right where the data is! Lets create a couple of machine learning models quickly using standard SQL dialect!

Key Advantage of BQML:

- Create ML model very quickly using simple and standard SQL statements
- Evaluate and review model performance
- Manipulate and transform data using SQL while creating a model

- Quick experiments to identify key features, Model behavior and predictive analysis
- Import and export models into Tensorflow

You can walk-through Jupyter notebook with detailed description of this example:

```
!gsutil cp gs://hla_peptide_dataset/notebooks/MHC_BigQuery_Demo.ipynb .
```

Using BQ WebUI, for our coronavirus subset of data, let's examine data first:

Following query shows information about antigen. Observe specific antigen that might

of more interest to our study for coronavirus, **can you find it?**

```
select * from `corona.antigen` limit 10
```

Row	Antigen_Name	Organism_Name	__Epitopes	__Assays	__References
1	Nucleoprotein	Avian coronavirus	1	2	1
2	Other Avian coronavirus protein	Avian coronavirus	1	1	1
3	Replicase polyprotein 1ab	Betacoronavirus 1	1	1	1
4	Non-structural protein 2a	Betacoronavirus 1	1	5	1
5	Replicase polyprotein 1ab	Alphacoronavirus 1	1	1	1
6	Nucleoprotein	Alphacoronavirus 1	1	1	1
7	Envelope small membrane protein	Murine coronavirus	16	32	1
8	Membrane protein	Murine coronavirus	44	91	1
9	Spike glycoprotein	Human coronavirus 229E	3	6	1
10	nucleoprotein	Middle East respiratory syndrome-related coronavirus	94	94	1

Following query shows epitope detail for those linear peptides whose parent protein

came from Spike glycoprotein, which is our interests to investigate for possible antigen

to protect against coronavirus

```
select Epitope_ID, Object_Type, Description, Starting_Position,
       Ending_Position, Antigen_Name, Parent_Protein, Organism_Name
from `corona.epitope`
where Parent_Protein = 'Spike glycoprotein'
```

Row	Epitope_ID	Object_Type	Description	Starting_Position	Ending_Position	Antigen_Name	Parent_Protein	Organism_Name
1	2801	Linear peptide	ALNTLVKQL	940	948	Spike glycoprotein precursor	Spike glycoprotein	SARS coronavirus BJ01
2	12967	Linear peptide	ELCDNPFFA	131	139	Spike glycoprotein precursor	Spike glycoprotein	SARS coronavirus BJ01
3	26710	Linear peptide	IITDNTFV	1096	1104	Spike glycoprotein precursor	Spike glycoprotein	SARS coronavirus BJ01
4	27146	Linear peptide	ILLCCMTSC	1214	1222	Spike glycoprotein precursor	Spike glycoprotein	SARS coronavirus BJ01
5	27241	Linear peptide	ILPDLKPT	787	795	Spike glycoprotein precursor	Spike glycoprotein	SARS coronavirus BJ01
6	32036	Linear peptide	KLNDLCFSNV	373	382	Spike glycoprotein precursor	Spike glycoprotein	SARS coronavirus BJ01
7	32068	Linear peptide	KLPDDFMGC	411	419	Spike glycoprotein precursor	Spike glycoprotein	SARS coronavirus BJ01
8	32069	Linear peptide	KLPDDFMGCV	411	420	Spike glycoprotein precursor	Spike glycoprotein	SARS coronavirus BJ01
9	36724	Linear peptide	LITGRLQSL	978	986	Spike glycoprotein precursor	Spike glycoprotein	SARS coronavirus BJ01

As you can see from the resulting query, we have a set of peptides generated from parent antigen protein. Now, we need to identify how these peptides bind with HLA molecule.

```
# Following query shows HLA-peptide binding affinity information. The goal is to
# leverage machine learning models to predict binding affinity for many different and
# new peptides with HLA MHC-Class-I molecules so that testing for vaccine candidates
# can be prioritized to accelerate solutions.
```

```
select Reference_ID, Description, Allele_Name, MHC_allele_class, Assay_Group,
       Qualitative_Measure, Quantitative_measurement
from `corona.mhc_ligand`
where Epitope_ID = 12967
```

Row	Reference_ID	Description	Allele_Name	MHC_allele_class	Assay_Group	Qualitative_Measure	Quantitative_measurement
1	642	ELCDNPFFA	HLA-A*02:01	I	qualitative binding	Positive	0.4

To build a machine learning model with BQ data, let's create a simple featured table, this is just one of the features that can be transformed for learning binding affinity. We will leverage amino_acid position in a given peptide as a feature to check if a specific sequence of amino acids in a peptide has good binding properties. There are many other features you can generate and use, such as amino acid polarity, weight, and generate many other cross-feature with data for HLA and peptides.

```
# Following query shows a featured table which will be used for BQML. This table is
# generated based on other base table with filter of only those linear peptides with
# length of 9 or 10 mers. The result shows 10 position for indexing amino acids in a
# given peptide as well as binding measures with quality and score.
```

```
select * from `corona.mhc_qual_feature`
```

Row	ligand_id	peptide	result_type	result	Allele	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	result_score	result_quality
1	1717014	ILLNKHID	Positive-Low	0	HLA-A*02:01	0	0	0	0	0	0	0	0	0	0	4000.0	1
2	1025027	AACAMLLVK	Negative	0	HLA-A*68:01	1	1	5	1	13	11	11	20	12	0	16800.0	0
3	1025025	AACAMLLVK	Negative	0	HLA-A*31:01	1	1	5	1	13	11	11	20	12	0	10600.0	0
4	1025026	AACAMLLVK	Negative	0	HLA-A*33:01	1	1	5	1	13	11	11	20	12	0	77700.0	0

Building ML models with BigQuery is as simple as writing SQL statements; makes ML modeling accessible to even SQL developers and analysts. We will create a simple classification model to predict for a given peptide if there is strong binding affinity with certain HLA Allele.

```
# Following statement creates a classification model using logistic regression. We are
# selecting feature columns of Allele and amino acid positioning within a peptide to
# classify if a peptide is a good candidate for vaccine testing.
```

Note: we are filtering data for peptides with length of 9 or 10 mers only. Also, since we
 # run multiple samples, we are randomizing samples by 80% of data for learning.

```
CREATE OR REPLACE MODEL `corona.Classification_model`
OPTIONS
(
  model_type='logistic_reg',
  input_label_cols=['result_quality']
)
AS
SELECT
  result_quality, Allele, A1, A2, A3, A4, A5, A6, A7, A8, A9, A10
FROM
  `corona.mhc_qual_feature`
WHERE length(peptide) IN (9,10)
AND rand() < 0.8;
```

Model will be created in a few min. Once created, you can review model stats by selecting a model within your dataset or by executing following query:

```
Select * from ML.EVALUATE(MODEL `corona.Classification_model`);
```

Row	precision	recall	accuracy	f1_score	log_loss	roc_auc
1	0.5324483131183888	0.3389211566370379	0.518096182449182	0.3300633060488028	1.607538922591955	0.7629848151848151

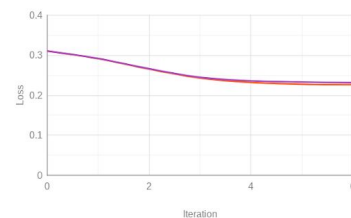
Classification_model

[QUERY MODEL](#)

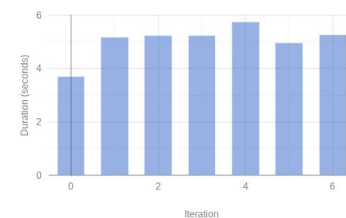
Details **Training** Evaluation Schema

View as **Graphs** Table

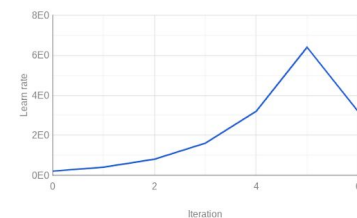
Loss



Duration (seconds)



Learn rate



Confusion matrix

This table shows the percentage of actual labels that were classified correctly (in blue) and incorrectly (in grey). The last column shows the percentage of total samples for the corresponding actual label.

Actual labels	Predicted labels					% samples
	0	1	2	3	4	
0	82.27%	5.2%	2.6%	9.93%	-	41.94%
1	57.69%	15.38%	6.21%	20.71%	-	16.76%
2	48.21%	11.61%	8.33%	31.85%	-	16.66%
3	28.63%	7.59%	5.86%	57.92%	-	22.86%
4	58.33%	-	-	36.11%	5.56%	1.78%

Based on your defined criteria, if the model needs improvement - do feature engineering and create a new model. For example, following criteria check quality of model we just created:

```
SELECT roc_auc, CASE
  WHEN roc_auc > .85 THEN 'good'
  WHEN roc_auc > .7 THEN 'fair'
  WHEN roc_auc > .5 THEN 'not great'
  ELSE 'poor' END AS model_quality
FROM ML.EVALUATE(MODEL `corona.Classification_model`);
```

Row	roc_auc	model_quality
1	0.7629848151848151	fair

BigQuery offers many [transform \(preprocessing\) functions](#) for feature engineering on data. Advantage of transform functions is that once you build a model with preprocessing as part of model definition, prediction data does not need to be prepared as the model will apply transformation for the input. Lets see one example of a transform feature and rebuild our model to check if we get better model performance.

```
# Following statement will create another classification model with preprocessing of the
# result score to normalize its deviation with respect to min-max value of an attribute.
CREATE OR REPLACE MODEL `corona.Classification_model2`
TRANSFORM ( result_quality, Allele, A1, A2, A3, A4, A5, A6, A7, A8, A9, A10,
            ML.MIN_MAX_SCALER(result_score) OVER() AS rs
          )
OPTIONS
(
  model_type='logistic_reg',
  input_label_cols=['result_quality']
)
AS SELECT result_quality, Allele, A1, A2, A3, A4, A5, A6, A7, A8, A9, A10, result_score
FROM `corona.mhc_qual_feature`
WHERE length(peptide) IN (9,10)
AND rand() < 0.8;
```

Lets evaluate the new model.

```
SELECT roc_auc, CASE WHEN roc_auc > .85 THEN 'good'
                     WHEN roc_auc > .7 THEN 'fair'
                     WHEN roc_auc > .5 THEN 'not great'
                     ELSE 'poor' END AS model_quality
FROM ML.EVALUATE(MODEL `corona.Classification_model2`);
```

Row	roc_auc	model_quality
1	0.883594005994006	good

With BigQuery you can take advantage of an already available highly powerful computer data processing and analysis platform to build a machine learning model, without moving your data! You can learn more about [BQML here](#). For our data set, you can build a DNN_Regression model to predict the result_score for a peptide HLA binding. Try that as a practice!

4. Accelerate with GCP AI Pipeline

For a data scientist, building ML models is highly focused on tasks such as feature engineering and model algorithms - mostly programming tasks with Python or so. In reality, operationalizing ML and AI requires end to end components from data sources to serving and monitoring ML models.

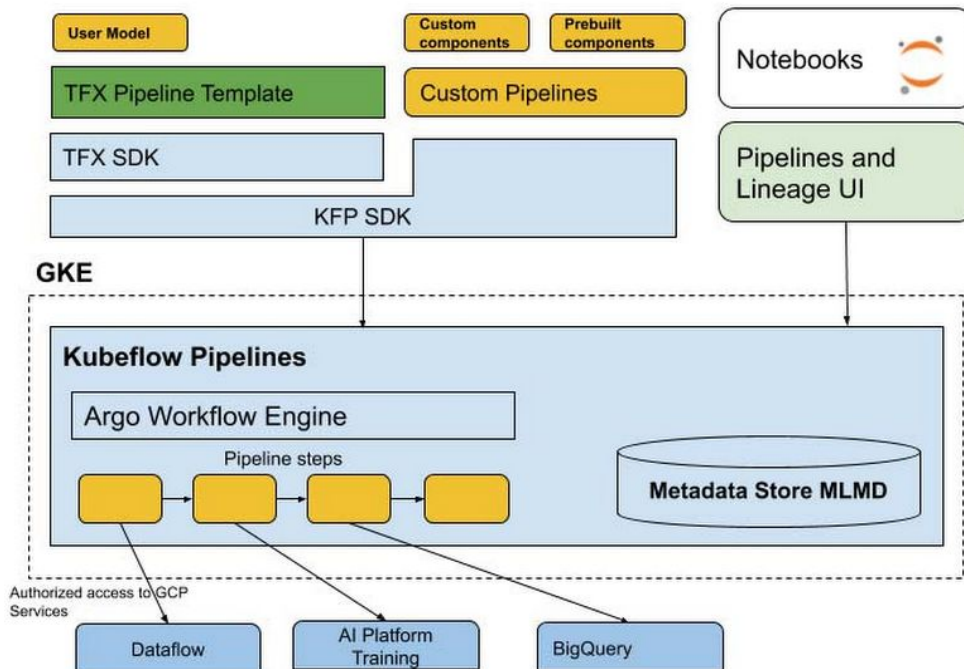
Google AI pipeline is a combination of Kubeflow Pipeline and Tensorflow Extension (TFX) framework that enables robust deployment of ML pipelines along with auditing and monitoring. As a part of Google AI Platform, AI pipeline enables developers to rapidly deploy multiple models and pipelines by leveraging reusable components of the pipeline.

Key benefits of GCP AI Pipeline:

- Push button installation for kubernetes cluster for ML Pipeline

- Enterprise ML deployment with logging and monitoring
- Automatic tracking of metadata and artifacts
- Reusable pipeline templates and pipeline components
- Integration with GCP services like BigQuery, Dataflow, AI Platform, and many others
- AI Pipeline [Documentation](#) and [Introduction Blog](#)

Architecture



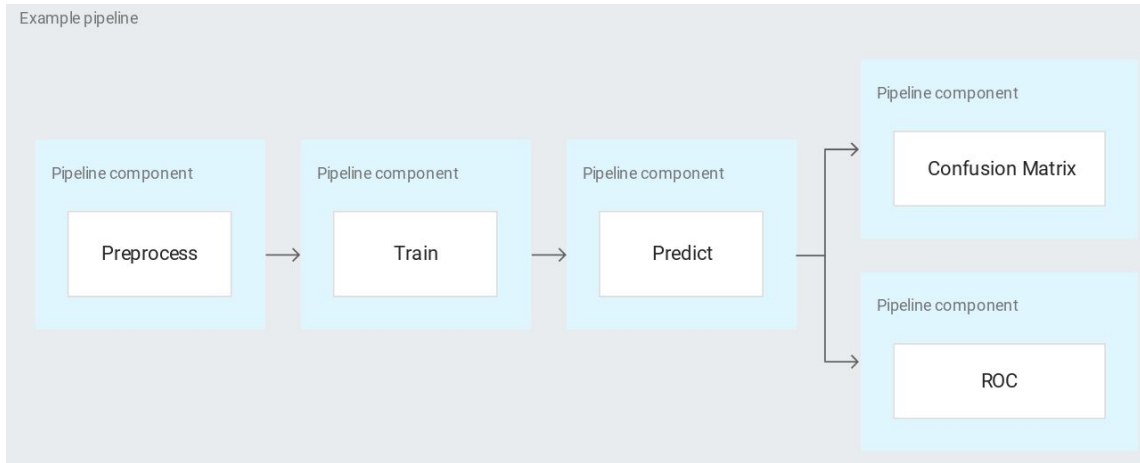
AI Platform Pipelines makes it easier to get started with MLOps by saving you the difficulty of setting up Kubeflow Pipelines with TensorFlow Extended (TFX). [Kubeflow Pipelines](#) is an open source platform for running, monitoring, auditing, and managing ML pipelines on Kubernetes. [TFX](#) is an open source project for building ML pipelines that orchestrate end-to-end ML workflows.

AI Pipeline Template

AI Platform Pipelines saves you the difficulty of:

- Setting up a [Google Kubernetes Engine](#) cluster
- Creating a [Cloud Storage](#) bucket
- Installing [Kubeflow Pipelines](#)

With AI Platform Pipelines, you can set up a Kubeflow Pipelines cluster in 15 minutes, so you can [quickly get started with ML pipelines](#). AI Platform Pipelines also creates a Cloud Storage bucket, to make it easier to run pipeline tutorials and [get started with TFX pipeline templates](#).



Lets deploy default AI pipeline templates that come with GCP AI Platform by setting up our Kubernetes cluster and Jupyter notebook environment.

From GCP Console → AI Platform → Pipelines , A AI Platform Pipelines page.

If you do not have any cluster created, click on New Instance to create a kubernetes cluster.

Google Cloud Platform

Covid-19

Search resources and products

AI Platform

Dashboard

AI Hub

Data Labeling

Pipelines

AI Platform Pipelines

BETA

NEW INSTANCE

REFRESH

DELETE

Filter

	Status	Name		Zone
		hosted-pipelines-quickstart	OPEN PIPELINES DASHBOARD	us-central1-a
		mhc-pipeline-container	OPEN PIPELINES DASHBOARD	us-central1-a

Click on **Configure** to continue.

Important: For a new cluster, select zone as: us-central1-a, make sure to **check** 'Allow access to Cloud APIs' box. Click Create Cluster. Wait for the cluster to be created, deploy the app instance.



Google Cloud Platform

Covid-19

Marketplace

Deploy Kubeflow Pipelines

Your app will use compute instances managed in a logical grouping called a "cluster", which will be configured in a way that's great for getting started with Kubernetes. For more options, visit the Kubernetes engine [cluster creation page](#).

Cluster zone ?

us-central1-a

☒ Allow access to the following Cloud APIs ?

<https://www.googleapis.com/auth/cloud-platform>

i All applications in cluster will have the access without additional settings. [Learn more](#) ↗

Create cluster

or [Select an existing cluster](#)

Namespace ?

default

App instance name ?

my-healthcare-pipelines

Deploy

You can see your pipeline cluster on AI Platform → Pipelines. Here is [documentation](#) for more detail on setting up AI Platform Pipelines. Settings of cluster will highlight cluster **endpoint url** that you would need to use within pipeline development in the next section.

Click on [OPEN PIPELINE DASHBOARD](#) from the Pipeline cluster information page.

Open [TF 2.1 Notebook](#) to see Pipeline template.

Getting Started

Pipelines

Experiments

Getting Started

Build your own pipeline with

TensorFlow Extended (TFX) [SDK](#) with end-to-end ML Pipeline Template ([Open TF 2.1 Notebook](#))

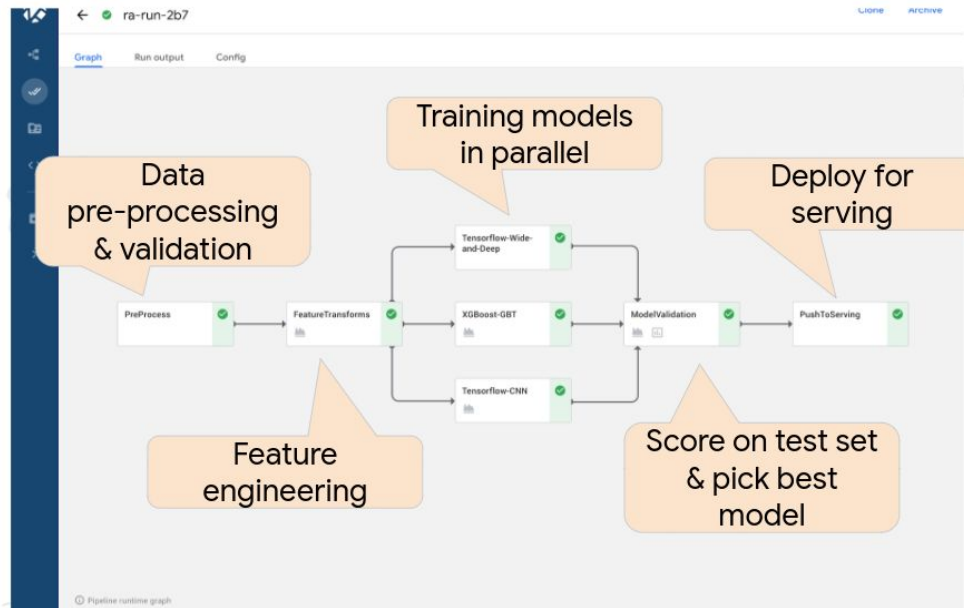
Kubeflow Pipelines [SDK](#)

You can continue to open a notebook instance created for you. You can create different notebook environments of your choice as needed. [Open JupyterLab](#).

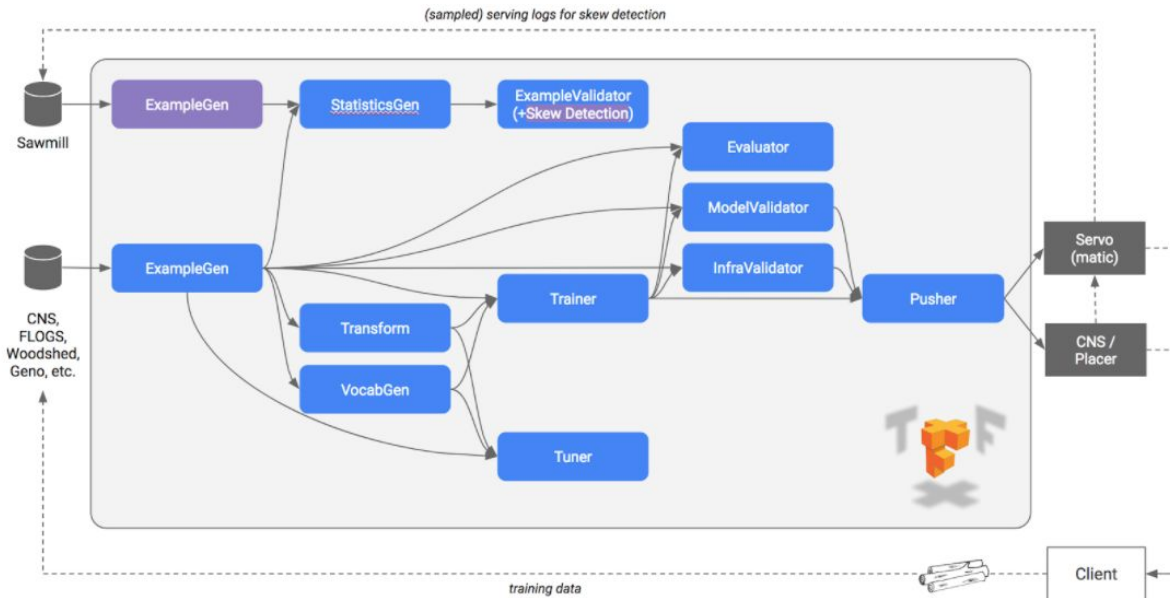
An example notebook of creating a TFX based ML pipeline is displayed from AI Hub to get you started quickly. You can review template instructions to deploy an example of ML model pipeline to predict taxi tips classification.

TFX template is organized with a set of reusable components that makes deploying ML model pipeline as easy as plug-and-play by replacing data source or model algorithm or feature set. Lets review template components. Let's look at a screenshot of an illustrative workflow that was run on Kubeflow Pipelines. This is just an illustrative workflow and users can author and run many different kinds of workflow topologies with different code&tools in the various steps of the workflow.

Visual depiction of pipeline topology



While using templates for Tensorflow Extension (TFX), one can quickly update/modify components to fit their use case and create a new pipeline. Following chart highlights an example of component flow from a design pipeline perspective.



The template pipeline provides the following code structure that you can quickly modify for your own ML application. We will be using the same template structure to deploy covid19 ML pipeline.

1. template.jpynb - notebook describes steps to set the environment and build a pipeline.
2. kubeflow_dag_runner.py - define runners for each orchestration engine - kubeflow.
3. pipeline.py - defines TFX components and a pipeline.
4. configs.py - defines common constants for pipeline runners
5. features.py - Define constants here that are common across all models including features names, label and size of vocabulary.
6. hparams.py - hyperparameters setting for model training performance.
7. model.py - tf.estimator for model definition leveraging features and parameters.
8. preprocessing.py - define your preprocessing of features.
9. In addition to above modularized pipeline structure, template also provides few utilities:
 - a. features_test.py - to write and test your features
 - b. model_test.py - test your model and evaluate
 - c. preprocessing_test.py - write and test preprocessing functions

You can test the default template by following notes within a notebook. For additional information on working with templates and sample pipelines, you can use [these examples](#).



Deploying Covid19 Pipelines Template

Make sure to have created the pipeline cluster as described in earlier sections and have launched the jupyterlab environment.

Create a new notebook under /AIHub/ folder and run the following command to bring demo/template notebook for peptide prediction.

```
!gsutil cp gs://hla_peptide_dataset/notebooks/Peptide_Prediction_Pipeline_Demo.ipynb .
```

```
[1]: !gsutil cp gs://hla_peptide_dataset/notebooks/Peptide_Prediction_Pipeline_Demo.ipynb .
      Copying gs://hla_peptide_dataset/notebooks/Peptide_Prediction_Pipeline_Demo.ipynb...
      / [1 files][ 27.0 KiB/ 27.0 KiB]
      Operation completed over 1 objects/27.0 KiB.
```

 Peptide_Prediction_Pipeline_Demo.ipynb

a pipeline notebook, copied under /AIHub/ folder.

You will have a notebook with detailed instructions to work through the peptide prediction pipeline. Let's examine, key sections of the notebook. Open a demo pipeline just copied and read through the instructions for each step as you progress:

Step 1: Setup your Environment

Execute commands to set versions of Tensorflow Extension (TFX) and validate setup. Since the notebook environment launched with python 3 already, ignore errors as described in the section. Key checkpoints:

- Validate TFX version
- Validate your GCP project name
- Must** update your kubeflow cluster ENDPOINT variable.

Step 2: Copy predefined pipeline template for peptide prediction

This step creates a working directory and copies required python files for a pipeline. Make sure to create a folder under /AIHub/ and run required commands. Key checkpoints:

- Provide a pipeline name, you can keep the default name for a demo.
- Make** sure to create a folder with same name as your pipeline name
- Run a command to bring template files into your demo folder
- Validate your current working directory

Step 3: Validate your template files

Brief introduction to each file and list files to validate, run a small test file. Key check points:

- List of files

Step 4: Run your Peptide Prediction Pipeline Demo

Update configuration file to reference to your GCS bucket where pipeline output and model export will be pushed to store. Key checkpoints:

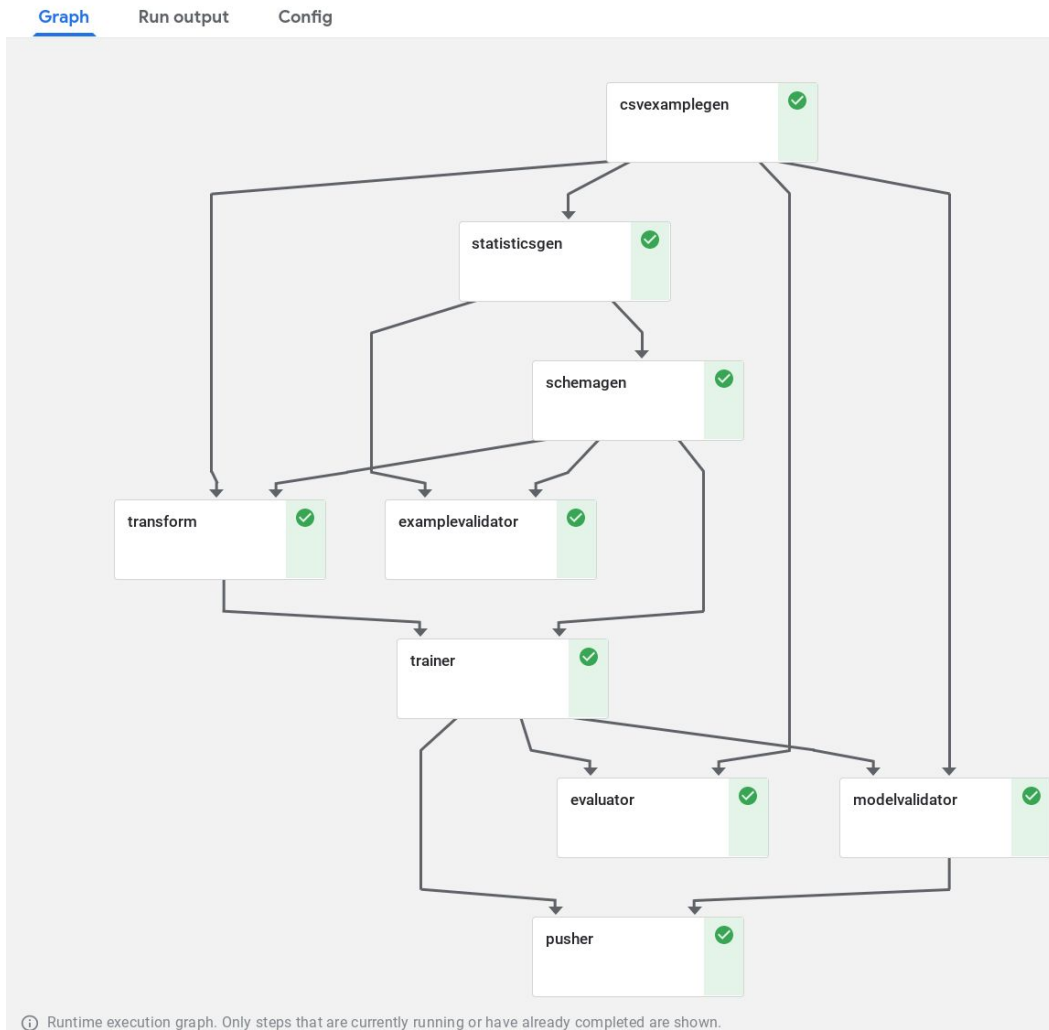
- Must** update GCS bucket name in a config file.
- Validate other variables in a config file
- Run and test your pipeline
- Check pipeline on pipeline dashboard of AI Platform

Pipelines			+ Upload pipeline	Refresh	Delete
Filter pipelines					
<input type="checkbox"/>	Pipeline name	Description	Uploaded on ↓		
<input type="checkbox"/>	▶ Peptide_Prediction_xx		4/7/2020, 9:39:14 AM		
<input type="checkbox"/>	▶ Peptide_Prediction		4/1/2020, 11:17:17 PM		

- Run pipeline experiment from notebook and review execution on a pipeline dashboard's experiment section. While the run job is executing, you can monitor pipeline experiment progress on the dashboard as well.

Experiments						
All experiments All runs						
Filter experiments						
Experiment name		Description		Last 5 runs		
▶ Peptide_Prediction_xx						
<input type="checkbox"/>	Run name	Status	Duration	Pipeline Version	Recurring ...	Start time
<input type="checkbox"/>	Peptide_Prediction_xx		-	Peptide_Prediction_xx	-	4/7/2020, 9:49:47 AM

Click on Experiment to see details of each step of the pipeline. Following chart displays full execution of pipeline, your template/demo pipeline has many components, which you can configure to add/remove by selecting them in a pipeline.py file.



Click on components such as statisticgen and evaluator to see details of data distribution and model performance.

Section 5 [optional]: Manage components of your pipeline

You can add/remove components for data validation including StatisticsGen, SchemaGen, and ExampleValidator. If you are interested in data validation, please see [Get started with Tensorflow Data Validation](#).

Section 6 [optional]: BigQueryExampleGen

Your demo pipeline is set up with sample data provided as a csv file. Practically, you would like to bring data from storage cloud or even more so from BigQuery where you may have large datasets. Use this section to learn more about how you can replace csvExampleGen components with BigQuery. Make sure to validate configuration requirements as described in

this section of notebook. You can leverage full epitope data sets from BigQuery's public dataset or from your own project. Key checkpoints:

- Update pipeline.py file with BQExampleGen instead of csv
- Make sure of project variables in a config file that has your dataset in BQ
- Make sure to validate BQ query arguments in a config file
- Make sure to update kubeflow runner file to enable query paraments
- Make sure to update features if you plan to use more or less of dataset attributes

Deploy Model to Serve Prediction

Section 7 [optional]: Deploy model for prediction

This section provides configuration leveraging AI platforms for training and then deploying a model for serving. Optionally, you can also deploy a model directly through the AI Platform Model dashboard. (**Internal Note:** [Bug](#) in process to fix api call to build model from notebook.)

Image below shows, running job on an AI platform that trains a model and deploys a model. This is particularly useful when you want a customized platform compute and large dataset to train models! Once the job is completed, you can see the model deployed on the model tab.

Job ID	Type	HyperTune	HyperTune parameters	Target metric	Create time	Elapsed time	Logs	Labels
tfx_20200407185736	Custom code training	No			Apr 7, 2020, 11:57:37 AM	2 min 21 sec	View Logs	tfx_execut...:tfx-compon...
tfx_20200402055644	Custom code training	No			Apr 1, 2020, 10:56:45 PM	6 min 22 sec	View Logs	tfx_execut...:tfx-compon...

By default, your model will be saved in your GCS bucket → tfx_pipeline_output folder. Lets deploy a version of the model to serve for prediction.

Click on **Create Model** on Model page of AI Platform dashboard.

Create model

Model Name *
Peptide_Prediction_Demo
Name is permanent, is case-sensitive, must start with a letter, and must only contain letters, numbers and underscores. Model names must be unique within each project. 23 / 128

Region *
us-central1
Service availability may vary based on region and model framework. See available regions and services for [TensorFlow](#) and [XGBoost](#)

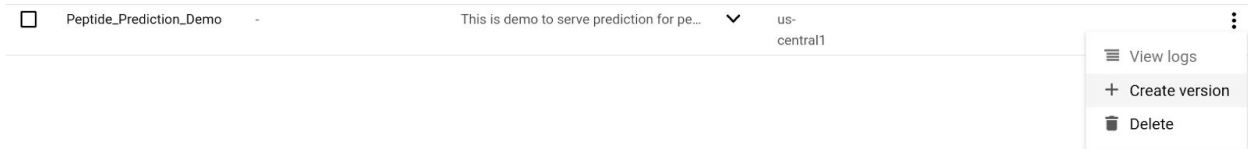
Description
This is demo to serve prediction for peptide binding

☐ Enable logging for this model ?
☐ Enable console logging for this model ?

Logging settings are permanent for this model. To change your logging preference in the future, create a new model.

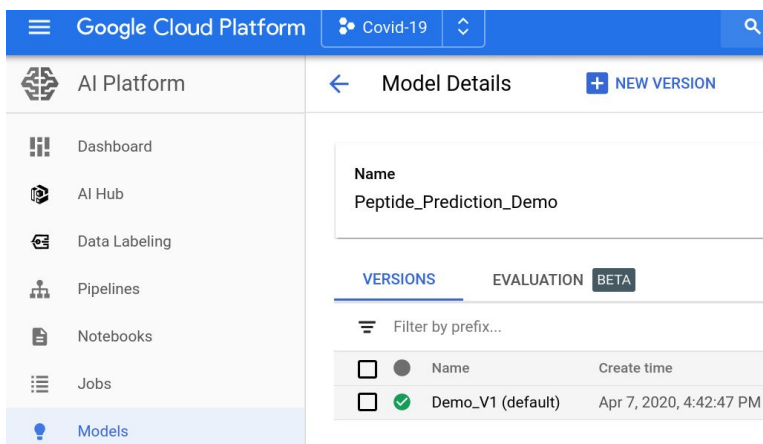
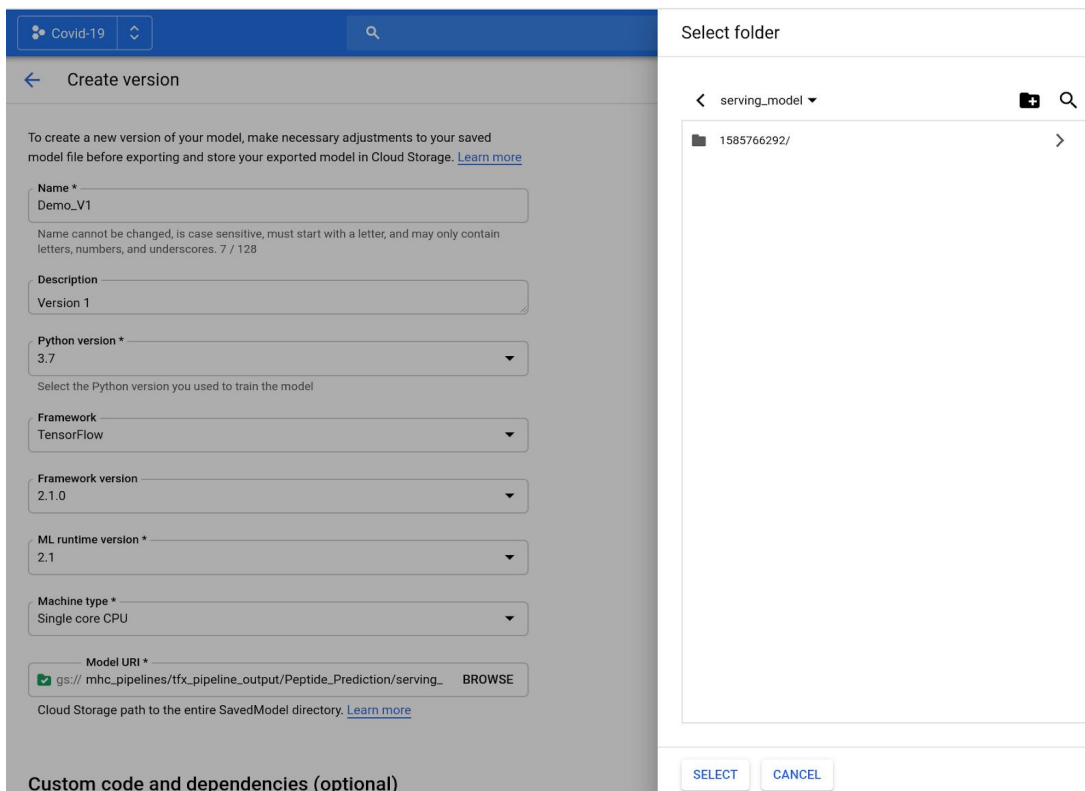
CREATE **CANCEL**

Click on create version of the model, here we can leverage saved models from GCS to be deployed as one of the versions of this model. You can deploy multiple versions as needed.



Make sure to select TF version 2.1 for framework and run time.

Make sure that to select a saved model, select parent dir from GCS from your model output dir
→ serving_dir → folder_name that contains saved_model.pb file.





Deploy the model, it is now ready to serve. Click on the deployed version, you can test it online or through the batch process. More details on the deployed model can be [found here](#).

Section 8 [optional]: Bring your own model and data to the pipeline

We made a pipeline for a model using the sample epitope dataset. Now it's time to put your data into the pipeline. Your data can be stored anywhere your pipeline can access, including GCS, or BigQuery. You will need to modify the pipeline definition to access your data. Key checkpoints:

- a. If your data is stored in files, modify the `DATA_PATH` in `kubeflow_dag_runner.py` or `beam_dag_runner.py` and set it to the location of your files. If your data is stored in BigQuery, modify `BIG_QUERY_QUERY` in `configs.py` to correctly query for your data.
- b. Add features in `features.py`.
- c. Modify `preprocessing.py` to [transform input data for training](#).
- d. Modify `model.py` and `hparams.py` to [describe your ML model](#).

Step 9: Cleaning Up Resources

To clean up all Google Cloud resources used in this demo project, you can [delete the Google Cloud project](#) you used for the tutorial.



Additional Examples

- a. For Pipeline operations and CI/CD, refer to [this example on github](#).
- b. [Custom model](#) with AI Platform
- c. **Predicting Heart Disease** ([Dataset](#) and [example](#))

Copy demo notebook by executing following command

```
!gsutil cp gs://hla_peptide_dataset/notebooks/Predict_Heart_Disease_Demo.ipynb .
```

Open Predict_Heart_Disease_Demo notebook, run through the steps, see details on each step in the above section for the peptide prediction pipeline. The key value of pipeline is that you can simply change your data set, model, features, processing as you need and re-deploy a model.

For example, for heart disease demo, we have simply changed following in a taxifare template pipeline:

- Pipeline name and pipeline files folder referenced in a notebook
- Data csv within data folder
- Features in feature.py file
- Validate settings in a config file for GCS project, bucket and pipeline name
- Create pipeline and deploy model just like a peptide example
-

- d. **Predicting Chronic Kidney Disease** ([Dataset](#))

One more example of quickly changing just data csv file and features + Label, we can create a new pipeline and model for predicting chronic kidney disease. Simply execute following command and step through the notebook: Predict_CKD_Disease_Demo.jpynb

```
!gsutil cp gs://hla_peptide_dataset/notebooks/Predict_CKD_Disease_Demo.jpynb .
```

Make sure to create a folder /AIHub/predict_ckd_demo. Validate config files for settings and create new pipelines, deploy a model and use it for prediction!

5.Future Development

We are working in integration with [Terra platform](#).

We continue to add more templates to provide the HCLS community with examples from different areas.

- [T Cell Receptor Specificity](#) - pipeline to illustrate use of AI/ML to predict how T cells recognize infected cells via the interactions between T cell receptors (TCRs) that are expressed on the surface of T cells and peptide-major histocompatibility complex (MHC) complexes that are expressed on the surface of target cells. The peptides that are presented by MHC molecules are termed T cell epitopes and are derived from pathogenic. Details of the model can be found in the [blog](#).
- [Predict protein- structure \(model\)](#) - pipeline for prediction protein structures associated with COVID-19. Details of the model can be found on DeepMind web site with the source code on [github](#).
- [Cancer Vaccine](#) - pipeline to illustrate use of AI/ML for selecting patient-specific cancer peptide vaccines. Details of the model can be found on [github](#).

6. References

A: Domain References

1. Predicting [HLA-A2 binding peptides](#)
2. New: Cov-19 - [Candidate Target for immune system](#)
3. [HLA Class consideration](#) for coronavirus
4. Peptides Database: <http://aps.unmc.edu/AP/main.php>
5. Peptides Calculator: http://aps.unmc.edu/AP/prediction/prediction_main.php
6. Peptides Database: <https://omictools.com/mhcpep-tool>
7. Peptides Database: <http://www.syfpeithi.de/0-Home.htm>
8. Protein Database:
<http://www.pdg.cnb.uam.es/cursos/Leon2002/pages/software/DatabasesListNAR2002/c>
[at/11](#)
Comments: [Know anti coronavirus peptides](#) which can inhibit some steps of virus development
9. [Cancer vaccine pipeline](#)
10. [Immune response to coronavirus](#)
11. [Potential vaccine](#) and immune response to Covid-19

B: Technical References

1. [Google AI Platform Pipelines](#)
2. [Peptide - HLA Binding CNN model](#)
3. [Epitope DB](#)
4. [MHC I binding prediction](#)
5. [MHC binding prediction in python](#)
6. [MHC binding prediction](#)
7. [TensorFlow for MHC binding prediction](#)
8. [Predicting for Chronic Kidney Disease](#)
9. [TF Tutorial for classification on structured data](#)
10. [Deep Learning model for Immune cell detecting Cancer Cell](#) but this approach can be reused for Immune cell detecting cell infected by virus.
11. [AI Pipeline Documentation](#)
12. [AI Platform Documentation](#)