**Overview**

In this lab you will predict the winner of a NCAA Men's Basketball tournament game using BigQuery, Machine Learning (ML), and the NCAA Men's Basketball dataset.

This lab uses [BigQuery Machine Learning](https://cloud.google.com/bigquery/docs/bigqueryml-analyst-start) (BQML), which allows you to use SQL to create ML models for classification and forecasting.

What you'll do

In this lab, you will learn how to:

* Use BigQuery to access the public NCAA dataset.
* Explore the NCAA dataset to gain familiarity with the schema and scope of the data available.
* Prepare and transform the existing data into features and labels.
* Split the dataset into training and evaluation subsets.
* Use BQML to build a model based on the NCAA tournament dataset.
* Use your newly created model to predict NCAA tournament winners for your bracket.

Prerequisites

This is a **fundamental level** lab. Before taking it, you should have some experience with SQL and the language's keywords. Familiarity with BigQuery is also recommended. If you need to get up to speed in these areas, you should at a minimum take one of the following labs before attempting this one:

* [Introduction to SQL for BigQuery and Cloud SQL](https://google.qwiklabs.com/catalog_lab/1282)
* [BigQuery: Qwik Start - Console](https://google.qwiklabs.com/catalog_lab/685)

Once you're ready, scroll down to learn about the services you will be using and how to properly set up your lab environment.

BigQuery

[BigQuery](https://cloud.google.com/bigquery/) is Google's fully managed, NoOps, low cost analytics database. With BigQuery you can query terabytes and terabytes of data without managing infrastructure or needing a database administrator. BigQuery uses SQL and takes advantage of the pay-as-you-go model. BigQuery allows you to focus on analyzing data to find meaningful insights.

There is a newly available dataset for NCAA basketball games, teams, and players. The game data covers play-by-play and box scores back to 2009, as well as final scores back to 1996. Additional data about wins and losses goes back to the 1894-5 season in some teams' cases.

Machine Learning

Google Cloud offers a [spectrum of Machine Learning options](https://cloud.google.com/products/#ai-and-machine-learning) for data analysts and data scientists. The most popular are:

* [Machine Learning APIs](https://cloud.google.com/products/ai/): use pretrained APIs like Cloud Vision for common ML tasks.
* [AutoML](https://cloud.google.com/automl/): create custom ML models with no coding needed.
* [BigQuery ML](https://cloud.google.com/bigquery/#bigqueryml): use your SQL knowledge to build ML models quickly right where your data already lives in BigQuery.
* [AI Platform](https://cloud.google.com/ml-engine/): build your own custom ML models and put them in production with Google's infrastructure.

In this lab you will use BigQuery ML to prototype, train, evaluate, and predict the 'winners' and 'losers' between two NCAA basketball tournament teams.

**Setup and requirements**

**Qwiklabs setup**

**Before you click the Start Lab button**

Read these instructions. Labs are timed and you cannot pause them. The timer, which starts when you click **Start Lab**, shows how long Google Cloud resources will be made available to you.

This Qwiklabs hands-on lab lets you do the lab activities yourself in a real cloud environment, not in a simulation or demo environment. It does so by giving you new, temporary credentials that you use to sign in and access Google Cloud for the duration of the lab.

**What you need**

To complete this lab, you need:

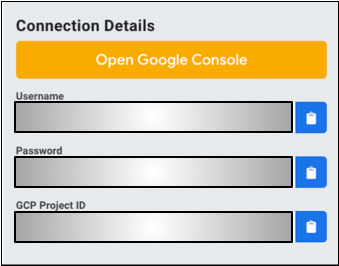
* Access to a standard internet browser (Chrome browser recommended).
* Time to complete the lab.

**Note:** If you already have your own personal Google Cloud account or project, do not use it for this lab.

**Note:** If you are using a Pixelbook, open an Incognito window to run this lab.

**How to start your lab and sign in to the Console**

1. Click the **Start Lab** button. If you need to pay for the lab, a pop-up opens for you to select your payment method. On the left, the **Connection Details** panel becomes populated with the temporary credentials that you must use for this lab.



1. Copy the username, and then click **Open Google Console**. The lab spins up resources, and then opens another tab that shows the **Choose an account** page.

***Tip:*** Open the tabs in separate windows, side-by-side.

1. On the Choose an account page, click **Use Another Account**.



1. The Sign in page opens. Paste the username that you copied from the Connection Details panel. Then copy and paste the password.

***Important:*** You must use the credentials from the Connection Details panel. Do not use your Qwiklabs credentials. If you have your own GCP account, do not use it for this lab (avoids incurring charges).

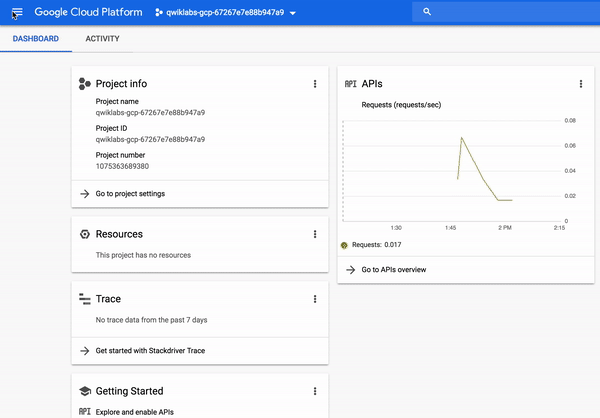
1. Click through the subsequent pages:
   * Accept the terms and conditions.
   * Do not add recovery options or two-factor authentication (because this is a temporary account).
   * Do not sign up for free trials.

After a few moments, the GCP console opens in this tab.

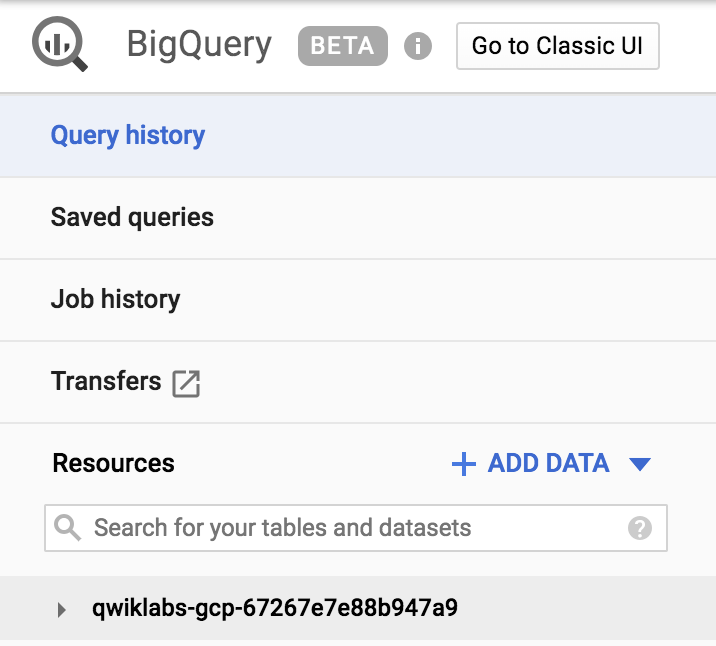
**Note:** You can view the menu with a list of GCP Products and Services by clicking the **Navigation menu** at the top-left, next to “Google Cloud Platform”. 

**Open the BigQuery Console**

From the Google Cloud Console, open the **Navigation menu** and select **BigQuery**:



Click the **Let's Go!** button to take yourself to the beta UI. Make sure that your Qwiklabs GCP Project ID is set in the left-hand Resources menu, which should resemble the following:



If you click the dropdown arrow next to your project, you won't see any databases or tables there. This is because you haven't added any to your project yet.

Luckily, there are tons of open, public datasets available in BigQuery for you to work with. You will now learn more about the NCAA dataset and then figure out how to add the dataset to your BigQuery project.

**NCAA March Madness**

The [National Collegiate Athletic Association](http://www.ncaa.org/) (NCAA) hosts two major college basketball tournaments every year in the United States for men's and women's collegiate basketball. For the NCAA Men's tournament in March, 68 teams enter single-elimination games and one team exits as the overall winner of March Madness.

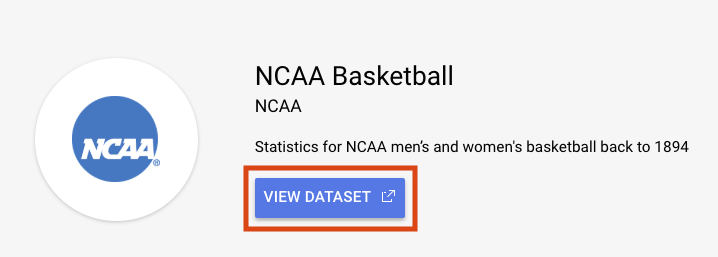
The NCAA offers a public dataset that contains the statistics for men's and women's basketball games and players in for the season and the final tournaments. The game data covers play-by-play and box scores back to 2009, as well as final scores back to 1996. Additional data about wins and losses goes back to the 1894-5 season in some teams' cases.

Be sure to check out the [Google Cloud Marketing Ad campaign for predicting live insights](https://www.google.com/amp/s/gweb-cloudblog-publish.appspot.com/products/gcp/architecting-live-ncaa-predictions-from-archives-to-insights/amp/) to learn a little bit more about this dataset and what's been done with it and stay up to date with this year's tournament at: [G.co/marchmadness](https://g.co/marchmadness)

**Find the NCAA public dataset in BigQuery**

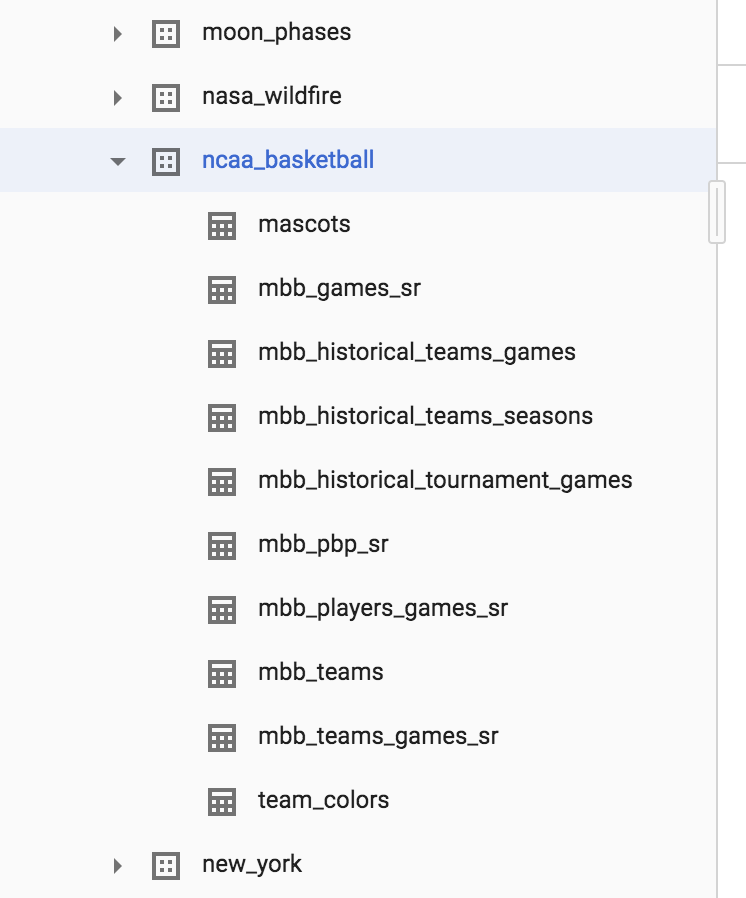
Make sure that you are still in the BigQuery Console for this step. Find the Resources tab from the left-hand menu and click the **+ ADD DATA** button then select **Explore public datasets**.

In the search bar, type in "NCAA Basketball" and hit enter. One result will pop up—select it and then click **VIEW DATASET**:

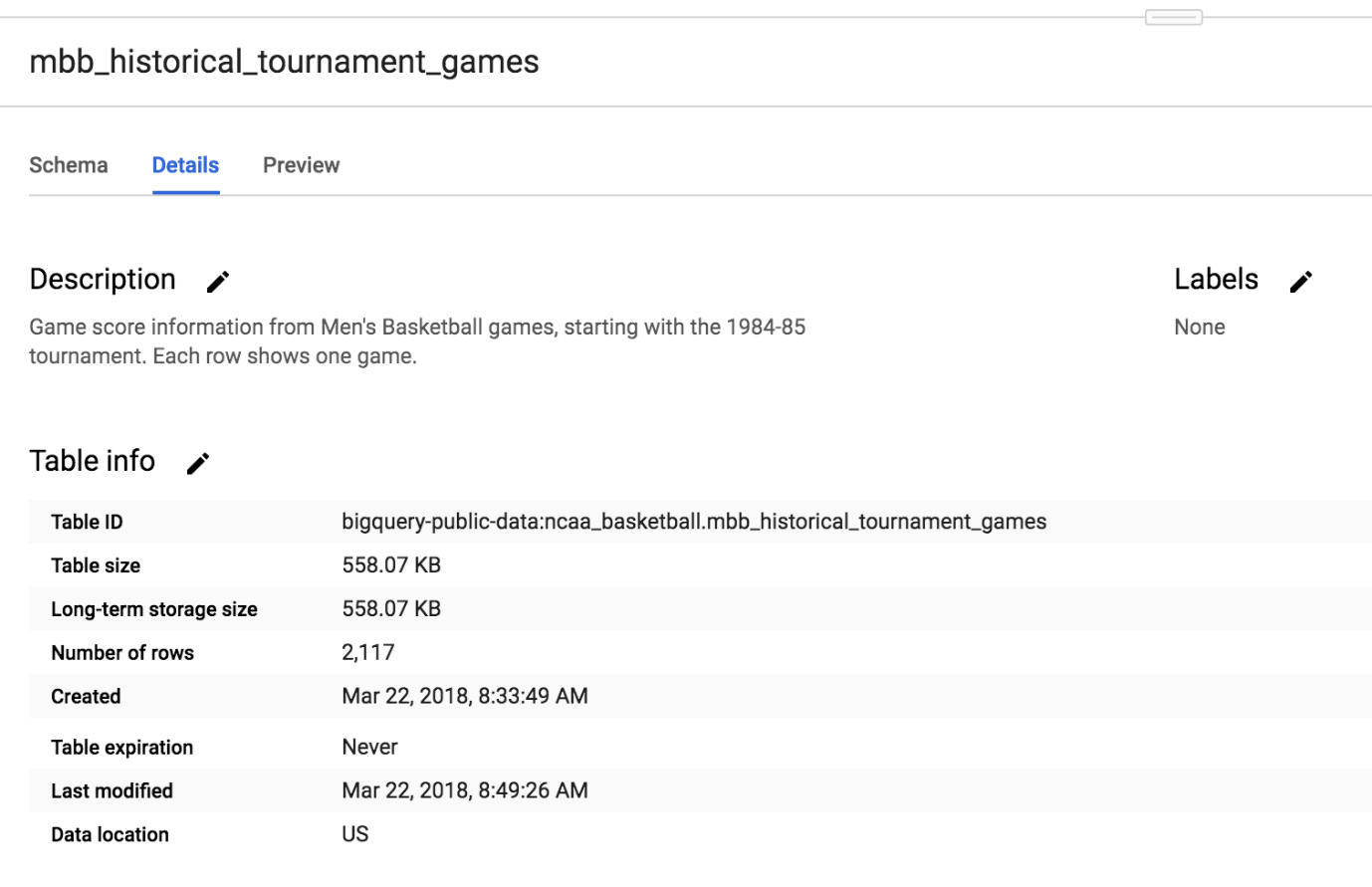


This will open a new BigQuery tab with the dataset loaded. You can continue working in this tab, or close it and refresh your BigQuery Console in the other tab to reveal your public dataset.

Click the arrow next to the ncaa\_basketball dataset to reveal its tables:



You should see 10 tables in the dataset. Click on the mbb\_historical\_tournament\_games and then click **Preview** to see sample rows of data. Then click **Details** to get metadata about the table. Your page should resemble the following:



**Write a query to determine available seasons and games**

You will now write a simple SQL query to determine how many seasons and games are available to explore in our mbb\_historical\_tournament\_games table.

Find the query editor, which is located above the table details section. Then copy and paste the following into that field:

SELECT

season,

COUNT(\*) as games\_per\_tournament

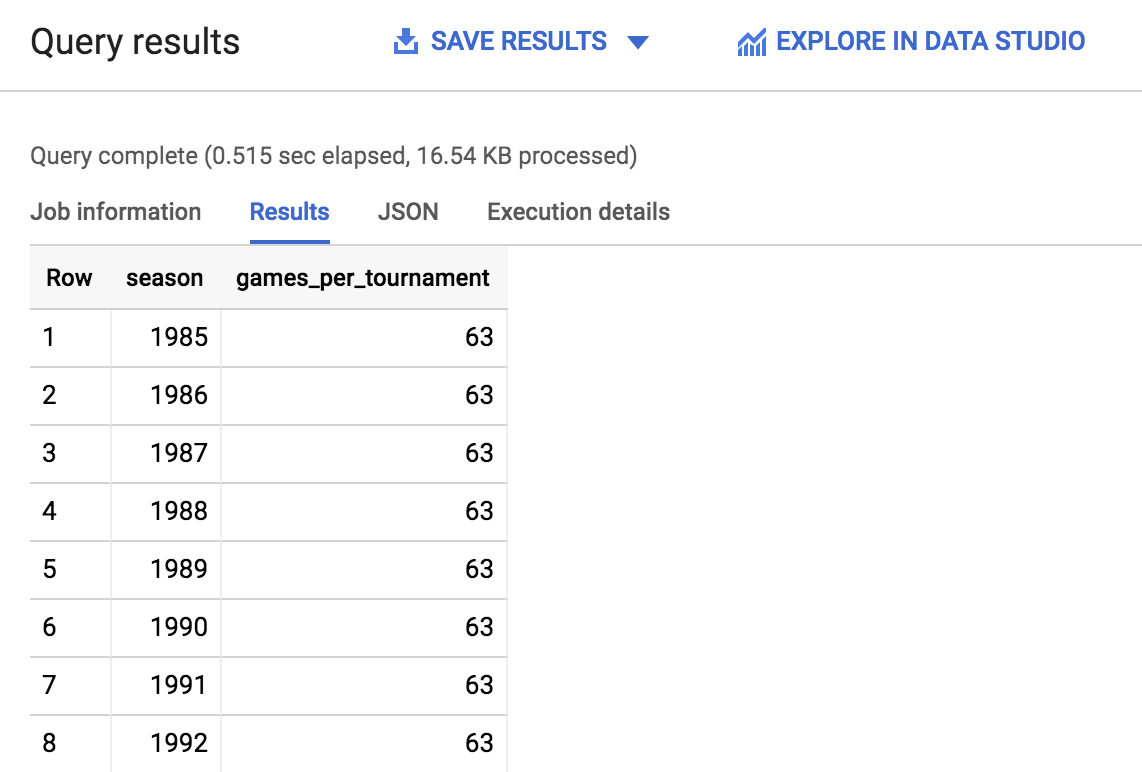
FROM

`bigquery-public-data.ncaa\_basketball.mbb\_historical\_tournament\_games`

GROUP BY season

ORDER BY season # default is Ascending (low to high)

Click **Run**. Soon after, you should receive a similar output:



Scroll through the output and take note of the amount of seasons and the games played per season—you will use that information to answer the following questions. Additionally, you can quickly see how many rows were returned by looking in the lower right near the pagination arrows.

Click *Check my progress* to verify the objective.

Write a query to determine available seasons and games

Check my progress

Test your understanding

The following multiple choice questions are used to reinforce your understanding of the concepts covered so far. Answer them to the best of your abilities.

How many NCAA Men's Basketball tournament seasons (i.e. 1985, 1986..) are available for us to explore?

close~~63~~

check33



67



70

Submit

How far back in time does the data in the NCAA Men's Basketball tournament data go for this table?



2017



1986

check1985



2019

Submit

**Understand machine learning features and labels**

The end goal of this lab is to predict the winner of a given NCAA Men's basketball match-up using past knowledge of historical games. In machine learning, each column of data that will help us determine the outcome (win or loss for a tournament game) is called a *feature*.

The column of data that you are trying to predict is called the *label*. Machine learning models "learn" the association between features to predict the outcome of a label.

Examples of features for your historical dataset could be:

* Season
* Team name
* Opponent team name
* Team seed (ranking)
* Opponent team seed

The label you will be trying to predict for future games will be the *game outcome*—whether or not a team wins or loses.

Test your understanding

The following multiple choice questions are used to reinforce your understanding of the concepts covered so far. Answer them to the best of your abilities.

For our historical training dataset, we know the correct answer for our labels (i.e. we know who won or lost)

checkTrue



False

Submit

For future predictions, we know the correct answer for our labels at the time of prediction.



True

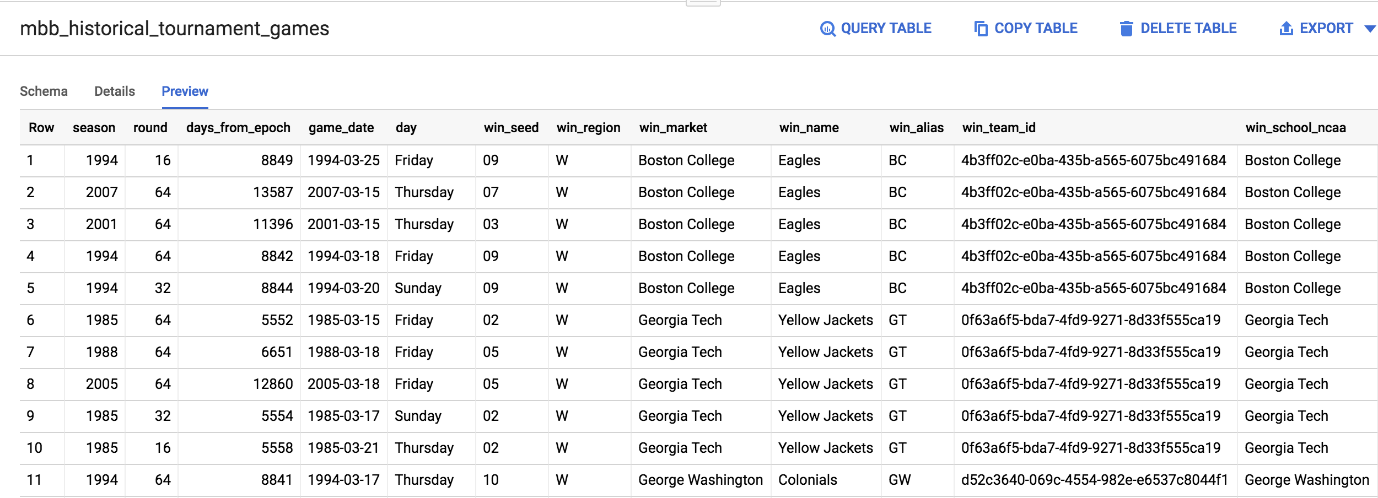
checkFalse

Submit

**Create a labeled machine learning dataset**

Building a machine learning model requires a lot of high-quality training data. Fortunately, our NCAA dataset is robust enough where we can rely upon it to build an effective model. Return to the BigQuery Console—you should have left off on the result of the query you ran.

From the left-hand menu, open the mbb\_historical\_tournament\_games table by clicking on the table name. Then once it loads, click **Preview**. Your page should resemble the following:



Test your understanding

The following multiple choice questions are used to reinforce your understanding of the concepts covered so far. Answer them to the best of your abilities.

Our labeled column for the outcome of the game already exists as a column in our dataset (i.e. a column that says WIN or LOSS)



True

checkFalse

Submit

For each specific basketball game, the dataset has one row for the winner and one row for the loser.

close~~True~~

checkFalse

Submit

After inspecting the dataset, you'll notice that one row in the dataset has columns for both win\_market and lose\_market. You need to break the single game record into a record for each team so you can label each row as a "winner" or "loser".

In the Query editor, copy and paste the following query and then click **Run**:

# create a row for the winning team

SELECT

# features

season, # ex: 2015 season has March 2016 tournament games

round, # sweet 16

days\_from\_epoch, # how old is the game

game\_date,

day, # Friday

'win' AS label, # our label

win\_seed AS seed, # ranking

win\_market AS market,

win\_name AS name,

win\_alias AS alias,

win\_school\_ncaa AS school\_ncaa,

# win\_pts AS points,

lose\_seed AS opponent\_seed, # ranking

lose\_market AS opponent\_market,

lose\_name AS opponent\_name,

lose\_alias AS opponent\_alias,

lose\_school\_ncaa AS opponent\_school\_ncaa

# lose\_pts AS opponent\_points

FROM `bigquery-public-data.ncaa\_basketball.mbb\_historical\_tournament\_games`

UNION ALL

# create a separate row for the losing team

SELECT

# features

season,

round,

days\_from\_epoch,

game\_date,

day,

'loss' AS label, # our label

lose\_seed AS seed, # ranking

lose\_market AS market,

lose\_name AS name,

lose\_alias AS alias,

lose\_school\_ncaa AS school\_ncaa,

# lose\_pts AS points,

win\_seed AS opponent\_seed, # ranking

win\_market AS opponent\_market,

win\_name AS opponent\_name,

win\_alias AS opponent\_alias,

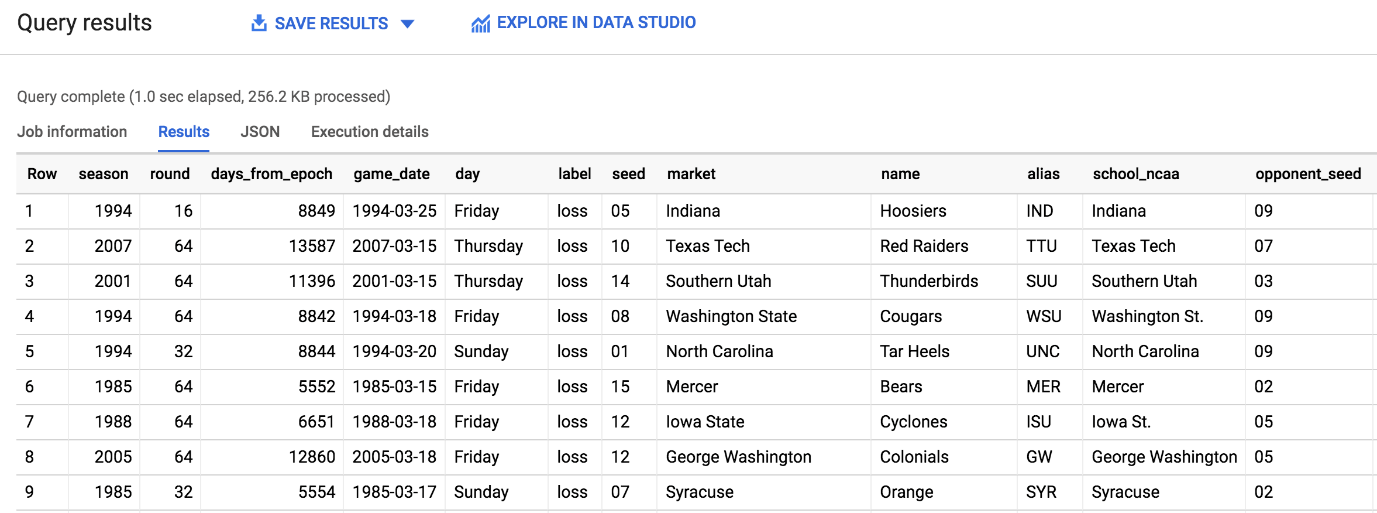
win\_school\_ncaa AS opponent\_school\_ncaa

# win\_pts AS opponent\_points

FROM

`bigquery-public-data.ncaa\_basketball.mbb\_historical\_tournament\_games`

You should receive the following output:



Click *Check my progress* to verify the objective.

Create a labeled machine learning dataset

Check my progress

Now that you know what features are available from the result, answer the following question to reinforce your understanding of the dataset.

Why should we NOT use the points scored (win\_pts or lose\_pts) as a feature in our training dataset if we have the data available?



This feature is only populated for recent games and should not be used.



This feature is numerical and cannot be used in a classification model.

checkThis feature is only available at the END of the game and for future games we are making predictions before a game begins.

Submit

**Part 1: Create a machine learning model to predict the winner based on seed and team name**

Now that we have explored our data, it's time to train a machine learning model. Using your best judgment, answer the question below to orient yourself with this section.

What type of machine learning model should we build knowing our goal is to predict game outcome (either win or lose)



We should build a forecasting model to predict how many points a team will score in a particular game.

checkWe should build a classification model to predict whether a team will win or lose a particular game.

Submit

Choosing a model type

For this particular problem, you will be building a classification model. Since we have two classes, win or lose, it's also called a binary classification model. A team can either win or lose a match.

If you wanted to, after the lab, you could forecast the total number of points a team will score using a forecasting model but that isn't going to be our focus here.

An easy way to tell if you're forecasting or classifying is to look at the type of label (column) of data you are predicting:

* If it's a numeric column (like units sold or points in a game), you're doing forecasting
* If it's a string value you're doing classification (this row is either this in class or this other class)
* ... and If you have more than two classes (like win, lose, or tie) you are doing multi-class classification

Our classification model will be doing machine learning is with a widely used statistical model called [Logistic Regression](https://en.wikipedia.org/wiki/Logistic_regression). We need a model that generates a probability for each possible discrete label value, which in our case is either a 'win' or a 'loss'. Logistic regression is a good model type to start with for this purpose. The good news for you is that the ML model will do all the math and optimization for you during model training -- it's what computers are really good at!

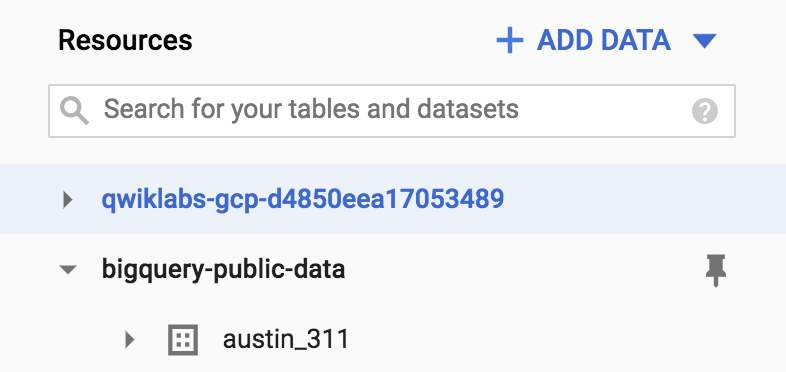
**Note:** There are many other machine learning models that vary in complexity to perform classification tasks. One commonly used at Google is [Deep Learning](https://en.wikipedia.org/wiki/Deep_learning) with [neural networks](https://playground.tensorflow.org/). You can learn the TensorFlow code needed to build neural networks in this [Data Science](https://www.qwiklabs.com/quests/28) quest.

Creating a machine learning model with BigQuery ML

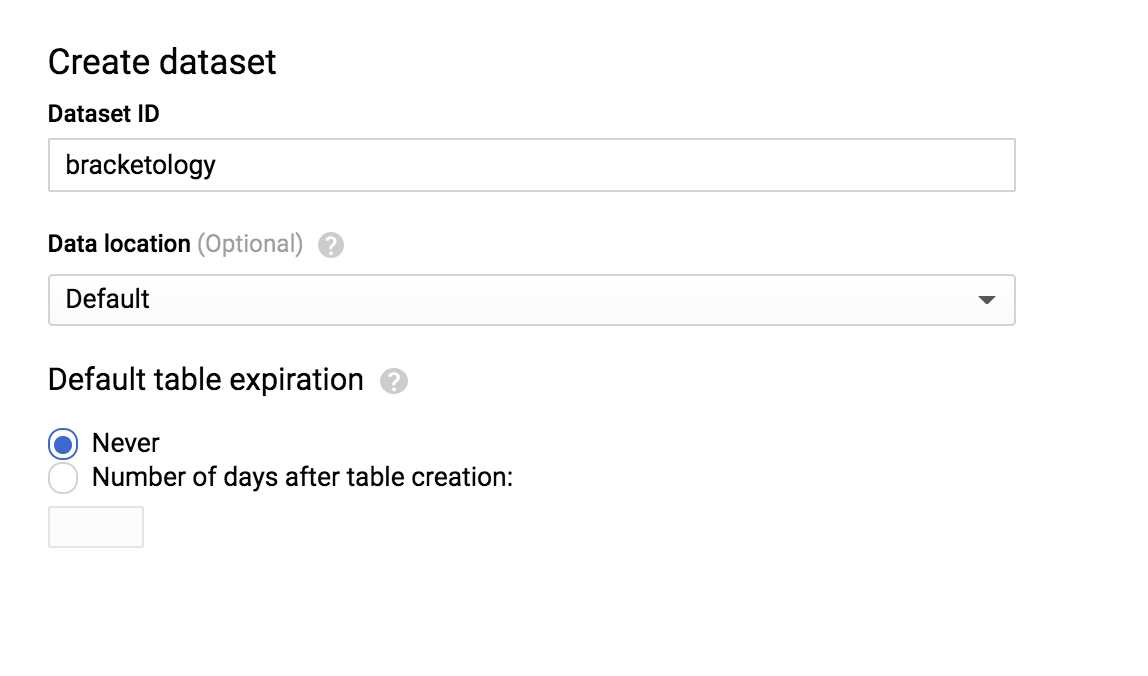
To create our classification model in BigQuery we simply need to write the SQL statement CREATE MODEL and provide a few options.

But, before we can create the model, we need a place in our project to store it first.

From the left-hand menu, select your Qwiklabs project from the Resources list:



Then from the right-hand side click **CREATE DATASET**. This will open a menu, set your Dataset ID to bracketology and click **Create dataset**:



Now run the following command in the Query editor

CREATE OR REPLACE MODEL

`bracketology.ncaa\_model`

OPTIONS

( model\_type='logistic\_reg') AS

# create a row for the winning team

SELECT

# features

season,

'win' AS label, # our label

win\_seed AS seed, # ranking

win\_school\_ncaa AS school\_ncaa,

lose\_seed AS opponent\_seed, # ranking

lose\_school\_ncaa AS opponent\_school\_ncaa

FROM `bigquery-public-data.ncaa\_basketball.mbb\_historical\_tournament\_games`

WHERE season <= 2017

UNION ALL

# create a separate row for the losing team

SELECT

# features

season,

'loss' AS label, # our label

lose\_seed AS seed, # ranking

lose\_school\_ncaa AS school\_ncaa,

win\_seed AS opponent\_seed, # ranking

win\_school\_ncaa AS opponent\_school\_ncaa

FROM

`bigquery-public-data.ncaa\_basketball.mbb\_historical\_tournament\_games`

# now we split our dataset with a WHERE clause so we can train on a subset of data and then evaluate and test the model's performance against a reserved subset so the model doesn't memorize or overfit to the training data.

# tournament season information from 1985 - 2017

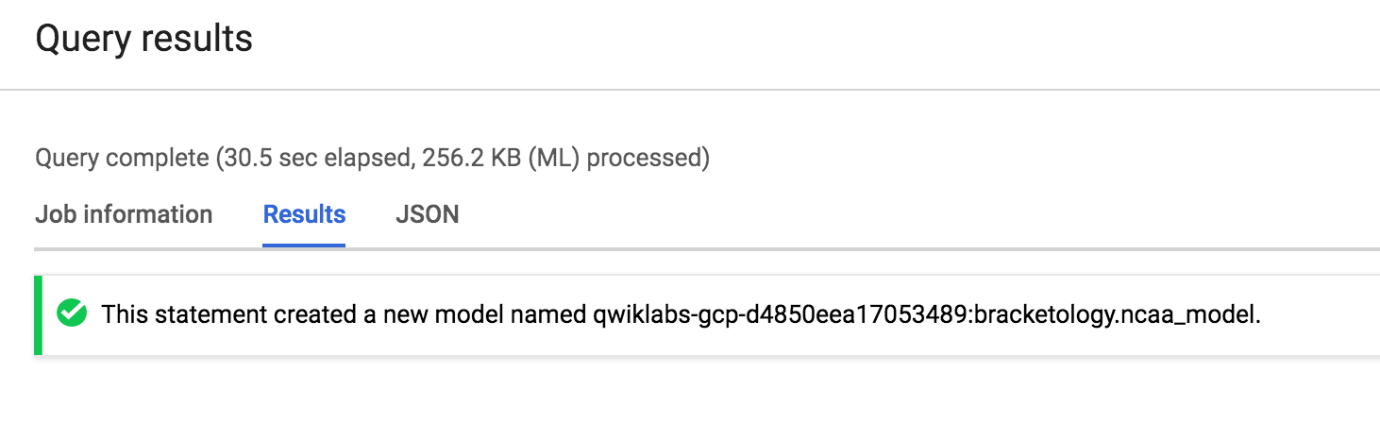
# here we'll train on 1985 - 2017 and predict for 2018

WHERE season <= 2017

In our code you'll notice that creating the model is just a few lines of SQL. One of the most important options is choosing the model type logistic\_reg for our classification task.

Refer to the BigQuery ML documentation guide for a list of all [available model options](https://cloud.google.com/bigquery/docs/reference/standard-sql/bigqueryml-syntax-create#model_option_list) and settings. In our case we already have a field named "label" so we avoid having to specify our label column by using the model option: input\_label\_cols.

It will take between 3 and 5 minutes to train the model. You should receive the following output when the job is finished:



Click the **Go to Model** button on the right-hand side of the console.

Click *Check my progress* to verify the objective.

Create a machine learning model

Check my progress

View model training details

Now that you are in the model details, scroll down to the **Training options** section and see the actual iterations the model performed to train. If you're experienced with machine learning, note that you can customize all of these hyperparameters (options set before the mdoel runs) by defining their value in the OPTIONS statement. If you're new to machine learning, BigQuery ML will set smart default values for any options not set.

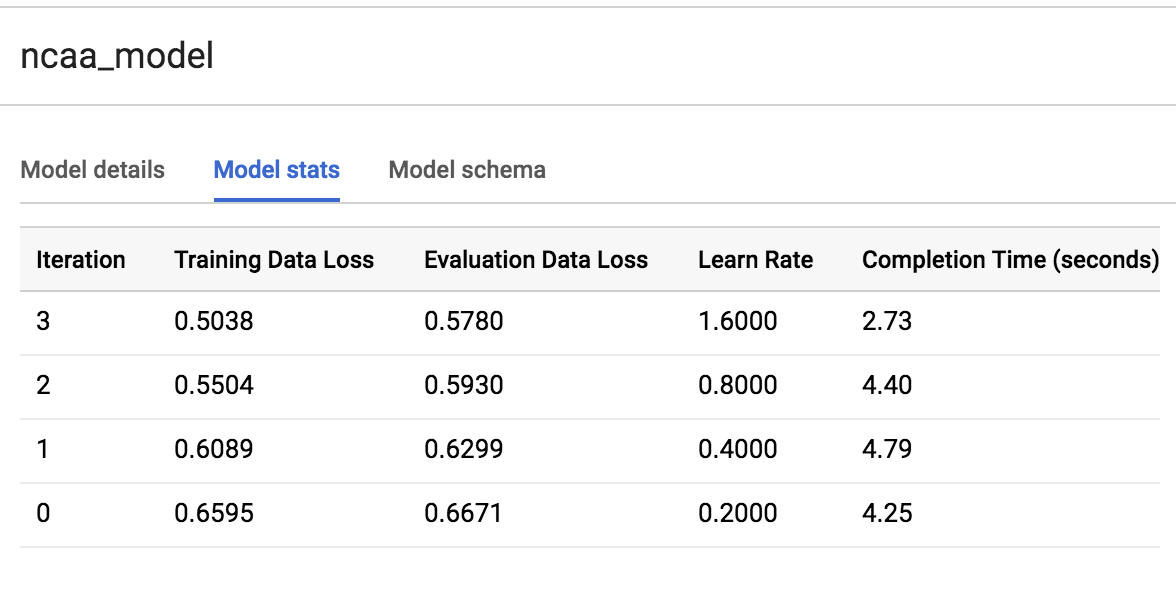
Refer to the [BigQuery ML model options list](https://cloud.google.com/bigquery/docs/reference/standard-sql/bigqueryml-syntax-create#model_option_list) to learn more.

View model training stats

Machine learning models "learn" the association between known features and unknown labels. As you might intuitively guess, some features like "ranking seed" or "school name" may help determine a win or loss more than other data columns (features) like the day of the week the game is played. Machine learning models start the training process with no such intuition and will generally randomize the weighting of each feature.

During the training process, the model will optimize a path to it's best possible weighting of each feature. With each run it is trying to minimize **Training Data Loss** and **Evaluation Data Loss**. Should you ever find that the final loss for evaluation is much higher than for training, your model is [overfitting](https://en.wikipedia.org/wiki/Overfitting) or memorizing your training data instead of learning generalizable relationships.

You can view how many training runs the model tooks by clicking the **Model Stats** or **Training** tab. During our particular run, the model completed 3 training iterations in about 20 seconds. Yours will likely vary.



See what the model learned about our features

After training, you can see which features provided the most value to the model by inspecting the weights. Run the following command in the Query editor:

SELECT

category,

weight

FROM

UNNEST((

SELECT

category\_weights

FROM

ML.WEIGHTS(MODEL `bracketology.ncaa\_model`)

WHERE

processed\_input = 'seed')) # try other features like 'school\_ncaa'

ORDER BY weight DESC

Your output should resemble the following:



As you can see, if the [seed](https://en.wikipedia.org/wiki/NCAA_basketball_tournament_selection_process#Seeding) of a team is very low (1,2,3) or very high (14,15,16) the model gives it a significant weight (max is 1.0) in determining the win loss outcome. Intuitively this makes sense as we expect very low seed teams to perform well in the tournament.

The real magic of machine learning is that we didn't create a ton of hardcoded IF THEN statements in SQL telling the model IF the seed is 1 THEN give the team a 80% more chance of winning. Machine learning does away with hardcoded rules and logic and learns these relationships for itself. Check out the [BQML syntax weights documentation](https://cloud.google.com/bigquery/docs/reference/standard-sql/bigqueryml-syntax-weights) to learn more.

**Evaluate model performance**

To evaluate the model's performance you can run a simple ML.EVALUATE against a trained model. Run the following command in the Query editor:

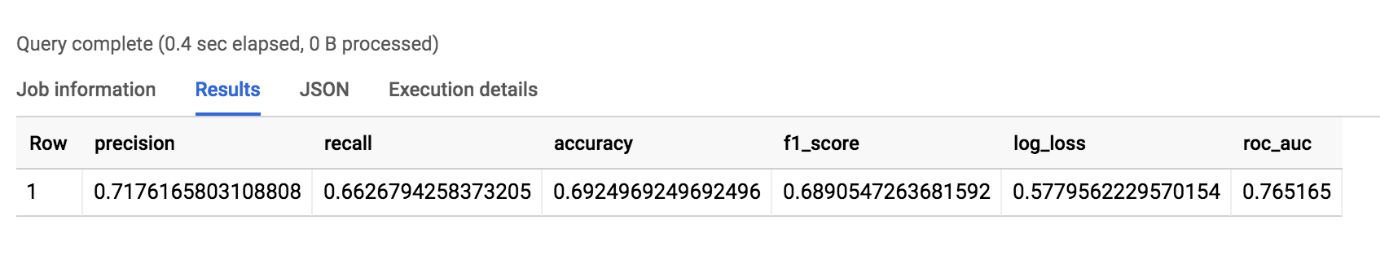
SELECT

\*

FROM

ML.EVALUATE(MODEL `bracketology.ncaa\_model`)

You should receive a similar output:



The value will be around 69% accurate. While it's better than a coin flip, there is room for improvement.

**Note:** For classification models, model accuracy isn't the only metric in the output you should care about. Because you performed a logistic regression, you can evaluate your model performance against all of the following metrics (the closer to 1.0 the better) **[Precision](https://developers.google.com/machine-learning/glossary/#precision)**: A metric for classification models. Precision identifies the frequency with which a model was correct when predicting the positive class. **[Recall](https://developers.google.com/machine-learning/glossary/#recall)**: A metric for classification models that answers the following question: Out of all the possible positive labels, how many did the model correctly identify? **[Accuracy](https://developers.google.com/machine-learning/glossary/#accuracy)**: Accuracy is the fraction of predictions that a classification model got right. **[f1\_score](https://en.wikipedia.org/wiki/F1\_score)**: A measure of the accuracy of the model. The f1 score is the harmonic average of the precision and recall. An f1 score's best value is 1. The worst value is 0. **[log\_loss](https://en.wikipedia.org/wiki/Cross\_entropy#Cross-entropy\_error\_function\_and\_logistic\_regression)**: The loss function used in a logistic regression. This is the measure of how far the model's predictions are from the correct labels. **[roc\_auc](https://developers.google.com/machine-learning/glossary/#AUC)**: The area under the ROC curve. This is the probability that a classifier is more confident that a randomly chosen positive example is actually positive than that a randomly chosen negative example is positive.

What do you think we could do to improve the model accuracy?

checkOur data only looks at NCAA tournament data which is a small part of the entire basketball season. If we included data on the regular season games the model may learn additional insights about the teams

checkExperiment with adding more features from our initial dataset and re-training the model (feature engineering).



Add back the total points scored in the tournament game to the training dataset

Submit

**Making predictions**

Now that you trained a model on historical data up to and including the 2017 season (which was all the data you had), it's time to make predictions for the 2018 season. Your data science team has just provided you with the tournament results for the 2018 tournament in a separate table which you don't have in your original dataset.

Making predictions is as simple as calling ML.PREDICT on a trained model and passing through the dataset you want to predict on.

Run the following command in the Query editor:

CREATE OR REPLACE TABLE `bracketology.predictions` AS (

SELECT \* FROM ML.PREDICT(MODEL `bracketology.ncaa\_model`,

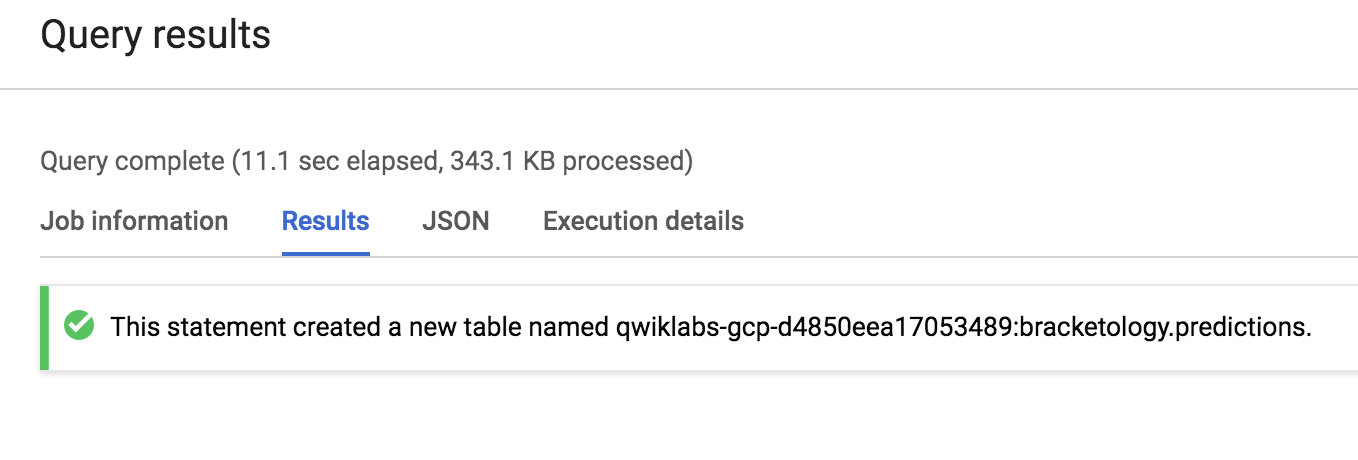
# predicting for 2018 tournament games (2017 season)

(SELECT \* FROM `data-to-insights.ncaa.2018\_tournament\_results`)

)

)

You should receive a similar output soon after:



Click *Check my progress* to verify the objective.

Evaluate model performance and create table

Check my progress

Why could we bring back the data columns for win\_pts, lose\_pts, game\_date, and round in our prediction? Isn't that not allowed?



You should not include these new columns in your prediction output.



The model is already trained. Using these new data columns now will only increase the accuracy of the prediction.

checkThe model is already trained. Adding these columns back during prediction-time provides additional context for the user but they do not get used as model features.

Submit

**Note**: You're storing your predictions in a table so you can query for insights later without having to keep re-running the above query.

You will now see your original dataset plus the addition of three new columns:

* Predicted label
* Predicted label options
* Predicted label probability

Since you happen to know the results of the 2018 March Madness tournament, let's see how the model did with it's predictions. (Tip: If you're predicting for this year's March Madness tournament, you would simply pass in a dataset with 2019 seeds and team names. Naturally, the label column will be empty as those games haven't been played yet -- that's what you're predicting!).

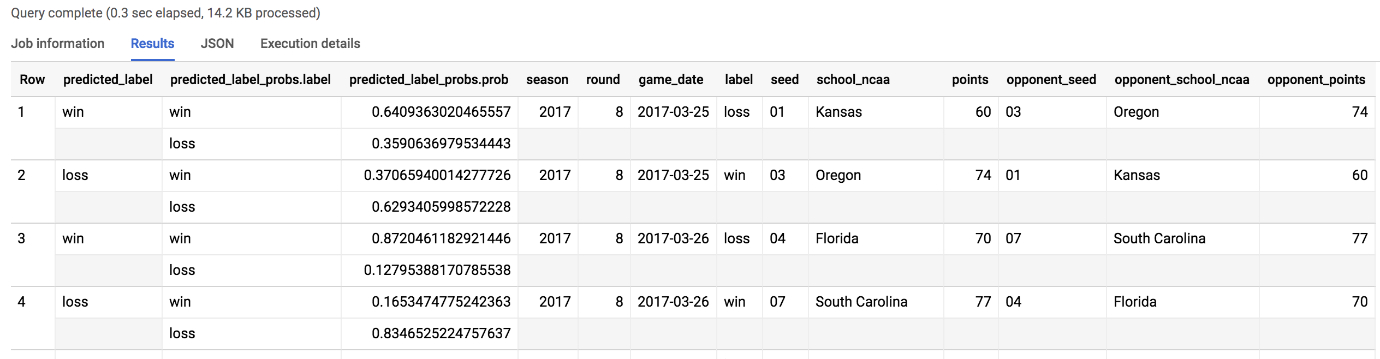
**How many did our model get right for the 2018 NCAA tournament?**

Run the following command in the Query editor:

SELECT \* FROM `bracketology.predictions`

WHERE predicted\_label <> label

You should receive a similar output:



Out of 134 predictions (67 March tournament games), our model got it wrong 38 times. 70% overall for the 2018 tournament matchup.

**Models can only take you so far...**

There are many other factors and features that go into the close wins and amazing upsets of any March Madness tournament that a model would have a very hard time predicting.

Let's find biggest upset for the 2017 tournament according to the model. We'll look where the model predicts with 80%+ confidence and gets it WRONG.

Run the following command in the Query editor:

SELECT

model.label AS predicted\_label,

model.prob AS confidence,

predictions.label AS correct\_label,

game\_date,

round,

seed,

school\_ncaa,

points,

opponent\_seed,

opponent\_school\_ncaa,

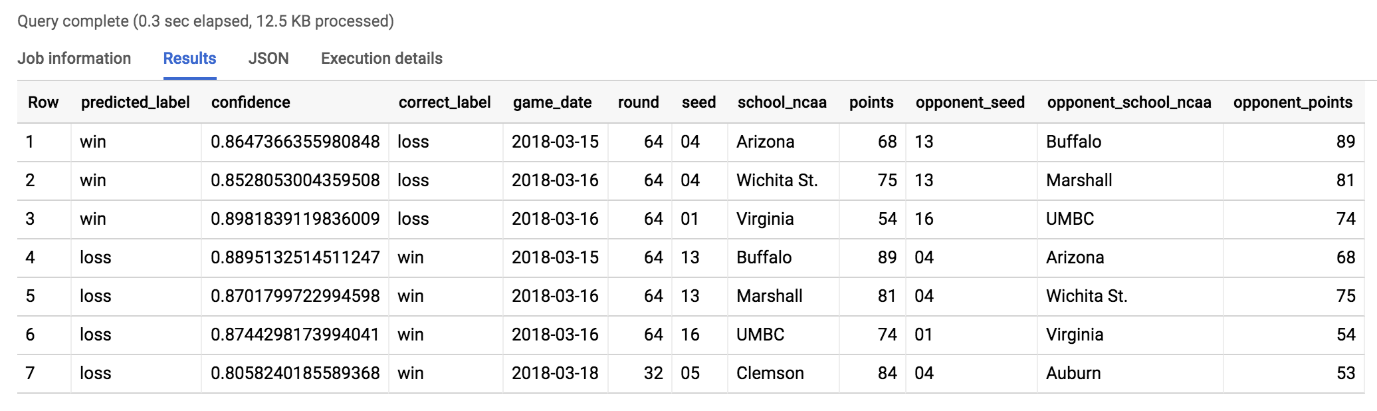
opponent\_points

FROM `bracketology.predictions` AS predictions,

UNNEST(predicted\_label\_probs) AS model

WHERE model.prob > .8 AND predicted\_label <> predictions.label

The outcome should look like this:



**Prediction**: The model predicts Seed 1 Virginia to beat Seed 16 UMBC with 87% confidence. Seems reasonable right?

Take a look at [this video](https://www.youtube.com/watch?v=-ckfno5hIIM) to see what actually happened!

Coach Odom (UMBC) after the [game](https://en.wikipedia.org/wiki/2018_UMBC_vs._Virginia_men%27s_basketball_game) said, "Unbelievable — it’s really all you can say."

Recap

* You created a machine learning model to predict game outcome.
* You evaluated the performance and got to 69% accuracy using seed and team name as primary features
* You predicted 2018 tournament outcomes
* You analyzed the results for insights

Our next challenge will be to build a better model WITHOUT using seed and team name as features.

**Part 2: Using skillful ML model features**

In the second part of this lab you will be building a second ML model using newly provided and detailed features.

What is a shortcoming of our first model?



The model heavily relied on team name as a feature. This punishes teams who had poor performance in previous years even if they have an all-star undefeated team this season.

close~~The model largely learned that lower seeded teams (1,2,3) would always beat higher seed teams (14,15,16).~~



The training dataset was limited to team name and seed. It ignored other key features like the pace or scoring efficiency of each team.

checkAll of the above

Submit

Now that you're familiar with building ML models using BigQuery ML, your data science team has provided you with a new play-by-play dataset where they have created new team metrics for your model to learn from. These include:

* Scoring efficiency over time based on historical play-by-play analysis.
* Possession of the basketball over time.

Create a new ML dataset with these skillful features

Run the following command in the Query editor:

# create training dataset:

# create a row for the winning team

CREATE OR REPLACE TABLE `bracketology.training\_new\_features` AS

WITH outcomes AS (

SELECT

# features

season, # 1994

'win' AS label, # our label

win\_seed AS seed, # ranking # this time without seed even

win\_school\_ncaa AS school\_ncaa,

lose\_seed AS opponent\_seed, # ranking

lose\_school\_ncaa AS opponent\_school\_ncaa

FROM `bigquery-public-data.ncaa\_basketball.mbb\_historical\_tournament\_games` t

WHERE season >= 2014

UNION ALL

# create a separate row for the losing team

SELECT

# features

season, # 1994

'loss' AS label, # our label

lose\_seed AS seed, # ranking

lose\_school\_ncaa AS school\_ncaa,

win\_seed AS opponent\_seed, # ranking

win\_school\_ncaa AS opponent\_school\_ncaa

FROM

`bigquery-public-data.ncaa\_basketball.mbb\_historical\_tournament\_games` t

WHERE season >= 2014

UNION ALL

# add in 2018 tournament game results not part of the public dataset:

SELECT

season,

label,

seed,

school\_ncaa,

opponent\_seed,

opponent\_school\_ncaa

FROM

`data-to-insights.ncaa.2018\_tournament\_results`

)

SELECT

o.season,

label,

# our team

seed,

school\_ncaa,

# new pace metrics (basketball possession)

team.pace\_rank,

team.poss\_40min,

team.pace\_rating,

# new efficiency metrics (scoring over time)

team.efficiency\_rank,

team.pts\_100poss,

team.efficiency\_rating,

# opposing team

opponent\_seed,

opponent\_school\_ncaa,

# new pace metrics (basketball possession)

opp.pace\_rank AS opp\_pace\_rank,

opp.poss\_40min AS opp\_poss\_40min,

opp.pace\_rating AS opp\_pace\_rating,

# new efficiency metrics (scoring over time)

opp.efficiency\_rank AS opp\_efficiency\_rank,

opp.pts\_100poss AS opp\_pts\_100poss,

opp.efficiency\_rating AS opp\_efficiency\_rating,

# a little feature engineering (take the difference in stats)

# new pace metrics (basketball possession)

opp.pace\_rank - team.pace\_rank AS pace\_rank\_diff,

opp.poss\_40min - team.poss\_40min AS pace\_stat\_diff,

opp.pace\_rating - team.pace\_rating AS pace\_rating\_diff,

# new efficiency metrics (scoring over time)

opp.efficiency\_rank - team.efficiency\_rank AS eff\_rank\_diff,

opp.pts\_100poss - team.pts\_100poss AS eff\_stat\_diff,

opp.efficiency\_rating - team.efficiency\_rating AS eff\_rating\_diff

FROM outcomes AS o

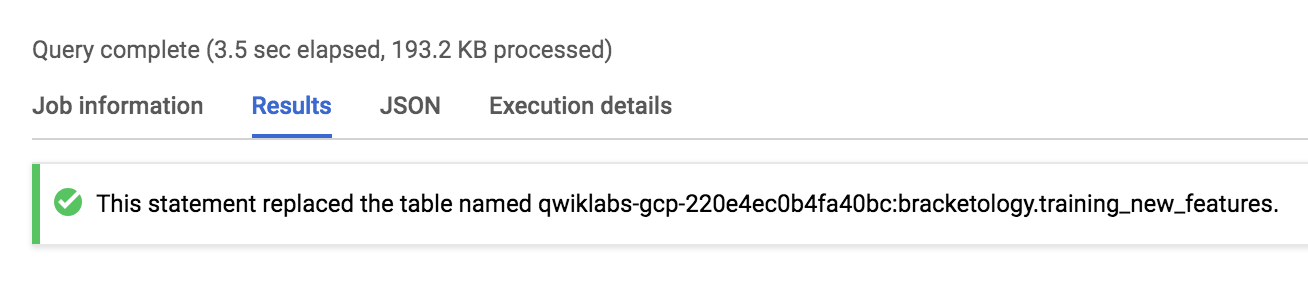
LEFT JOIN `data-to-insights.ncaa.feature\_engineering` AS team

ON o.school\_ncaa = team.team AND o.season = team.season

LEFT JOIN `data-to-insights.ncaa.feature\_engineering` AS opp

ON o.opponent\_school\_ncaa = opp.team AND o.season = opp.season

You should receive a similar output soon after:



Click *Check my progress* to verify the objective.

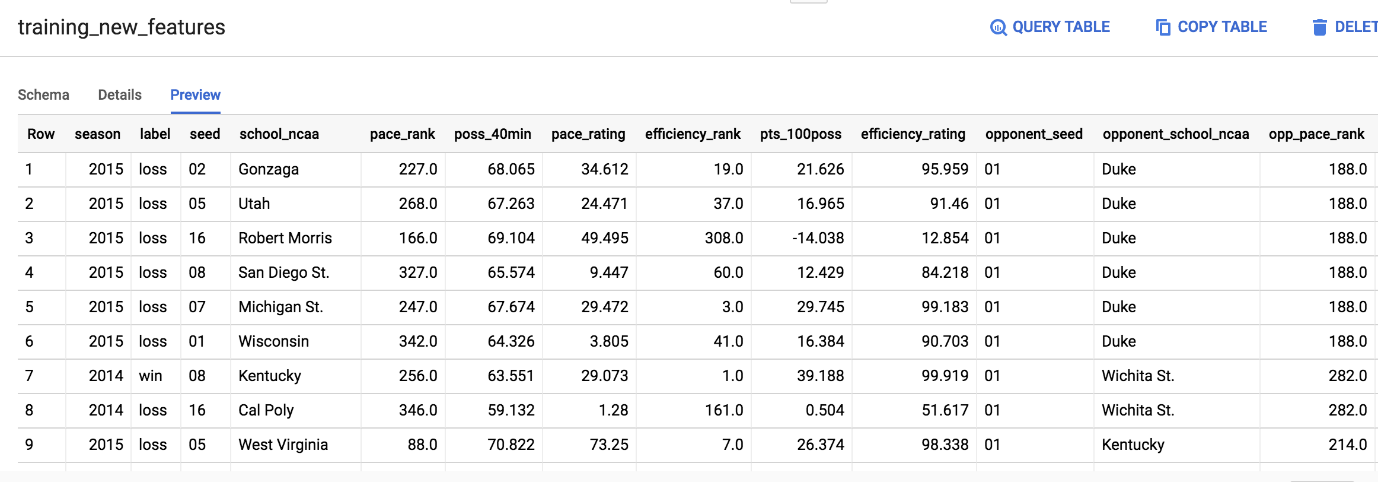
Using skillful ML model features

Check my progress

**Preview the new features**

Click on the **Go to table** button on the right-hand side of the console. Then click on the **Preview** tab.

Your table should look similar to the following:



Don't worry if your output is not identical to the above screenshot.

**Interpreting selected metrics**

You will now learn about some important labels that aid us in making predictions.

opp\_efficiency\_rank

**Opponent's Efficiency Rank**: out of all the teams, what rank does our opponent have for scoring efficiently over time (points per 100 basketball possessions). Lower is better.

opp\_pace\_rank

**Opponent's Pace Rank**: out of all teams, what rank does our opponent have for basketball possession (number of possessions in 40 minutes). Lower is better.

Now that you have insightful features on how well a team can score and how well it can hold on to the basketball lets train our second model.

As an additional measure to safe-guard your model from "memorizing good teams from the past", exclude the team's name and the seed from this next model and focus only on the metrics.

**Train the new model**

Run the following command in the Query editor:

CREATE OR REPLACE MODEL

`bracketology.ncaa\_model\_updated`

OPTIONS

( model\_type='logistic\_reg') AS

SELECT

# this time, dont train the model on school name or seed

season,

label,

# our pace

poss\_40min,

pace\_rank,

pace\_rating,

# opponent pace

opp\_poss\_40min,

opp\_pace\_rank,

opp\_pace\_rating,

# difference in pace

pace\_rank\_diff,

pace\_stat\_diff,

pace\_rating\_diff,

# our efficiency

pts\_100poss,

efficiency\_rank,

efficiency\_rating,

# opponent efficiency

opp\_pts\_100poss,

opp\_efficiency\_rank,

opp\_efficiency\_rating,

# difference in efficiency

eff\_rank\_diff,

eff\_stat\_diff,

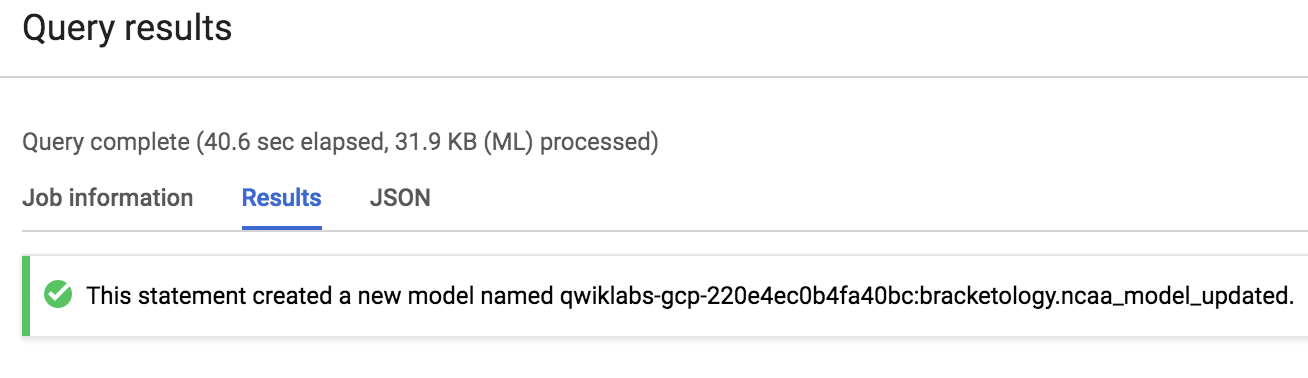
eff\_rating\_diff

FROM `bracketology.training\_new\_features`

# here we'll train on 2014 - 2017 and predict on 2018

WHERE season BETWEEN 2014 AND 2017 # between in SQL is inclusive of end points

Soon after, your output should look similar to the following:



**Evaluate the new model's performance**

To evaluate your model's performance, run the following command in the Query editor:

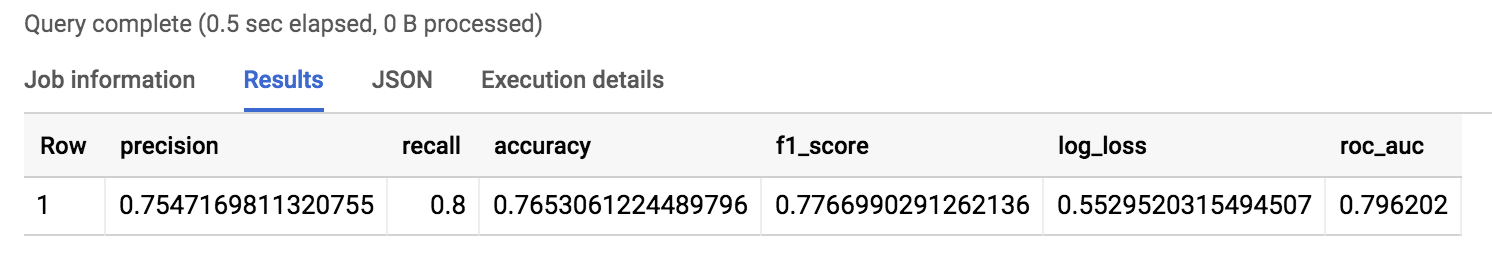
SELECT

\*

FROM

ML.EVALUATE(MODEL `bracketology.ncaa\_model\_updated`)

Your output should look similar to the followings:



Wow! You just trained a new model with different features and increased your accuracy to around 75% or a 5% lift from the original model.

This is one of the biggest lessons learned in machine learning is that your high quality feature dataset can make a huge difference in how accurate your model is.

Click *Check my progress* to verify the objective.

Train the new model and make evalution

Check my progress

**Inspect what the model learned**

Which features does the model weigh the most in win / loss outcome? Find out by running the following command in the Query editor:

SELECT

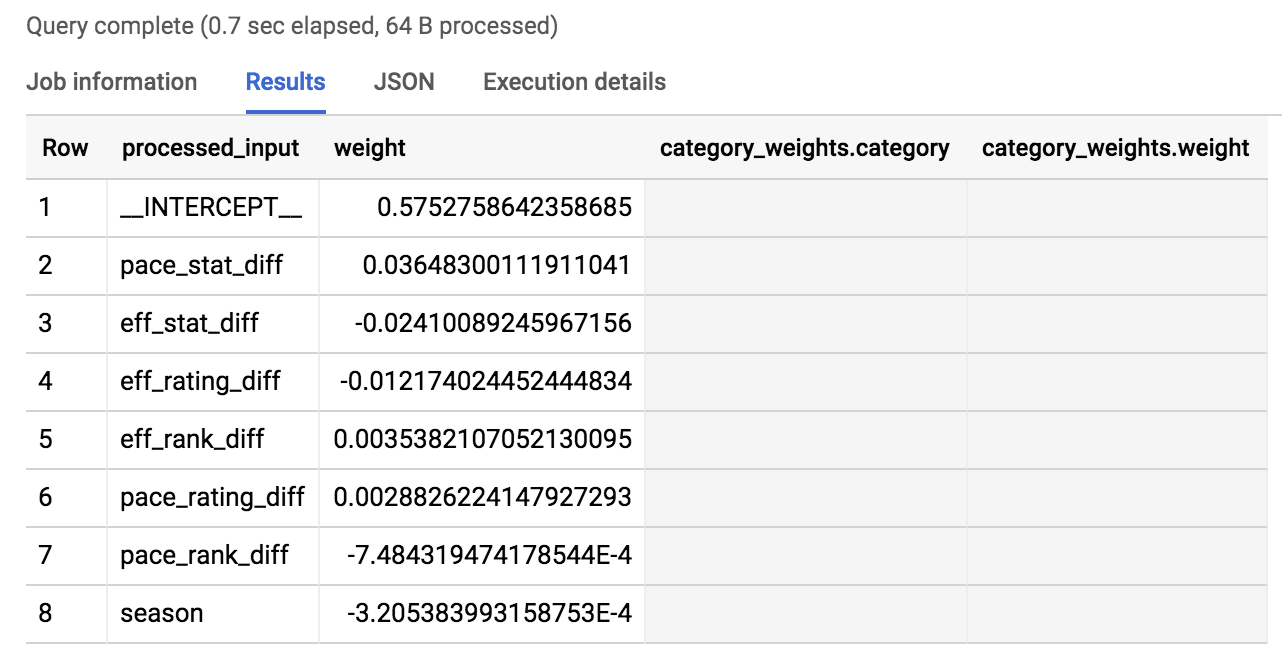
\*

FROM

ML.WEIGHTS(MODEL `bracketology.ncaa\_model\_updated`)

ORDER BY ABS(weight) DESC

Your output should look like this:



We've taken the absolute value of the weights in our ordering so the most impactful (for a win or a loss) are listed first.

As you can see in the results, the top 3 are pace\_stat\_diff, eff\_stat\_diff, and eff\_rating\_diff. Let's explore these a little bit more.

pace\_stat\_diff

How different the actual stat for (possessions / 40 minutes) was between teams. According to the model, this is the largest driver in choosing the game outcome.

eff\_stat\_diff

How different the actual stat for (net points / 100 possessions) was between teams.

eff\_rating\_diff

How different the normalized rating for scoring efficiency was between teams.

What did the model not weigh heavily in its predictions? Season. It was last in the output of ordered weights above. What the model is saying is that the season (2013, 2014, 2015) isn't that useful in predicting game outcome. There wasn't anything magical about the year "2014" for all teams.

An interesting insight is that the model valued the pace of a team (how well they can control the ball) over how efficiently a team can score.

**Prediction time!**

Run the following command in the Query editor:

CREATE OR REPLACE TABLE `bracketology.ncaa\_2018\_predictions` AS

# let's add back our other data columns for context

SELECT

\*

FROM

ML.PREDICT(MODEL `bracketology.ncaa\_model\_updated`, (

SELECT

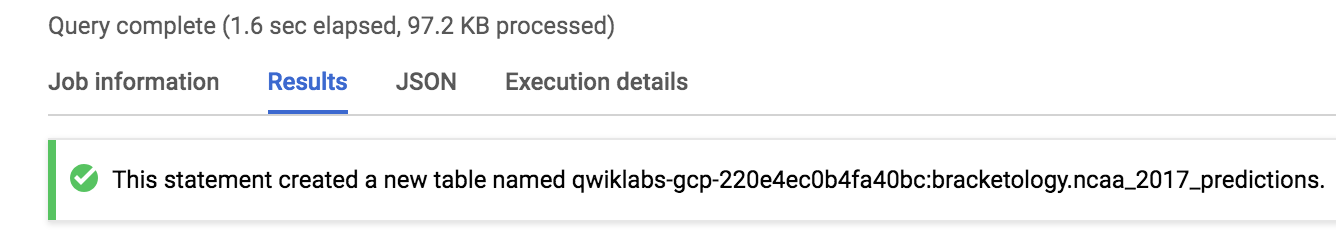
\* # include all columns now (the model has already been trained)

FROM `bracketology.training\_new\_features`

WHERE season = 2018

))

Your output should be similar to the following:



Click *Check my progress* to verify the objective.

Run a query to create a table ncaa\_2018\_predictions

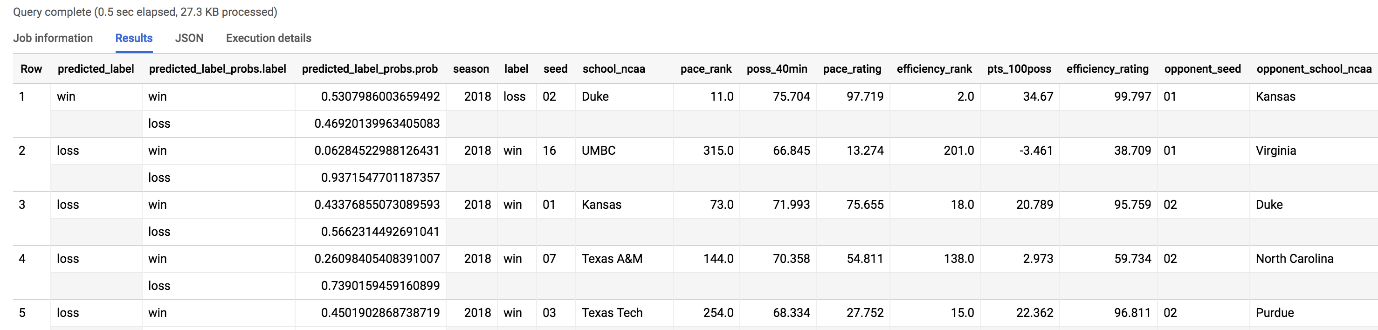
Check my progress

**Prediction analysis:**

Since you know the correct game outcome, you can see where your model made an incorrect prediction using the new testing dataset. Run the following command in the Query editor:

SELECT \* FROM `bracketology.ncaa\_2018\_predictions`

WHERE predicted\_label <> label



As you can see from the number of records returned from the query, the model got 48 matchups wrong (24 games) out of the total number of matchups in the tournament for a 2018 accuracy of 64%. 2018 must have been a wild year, let's look at what upsets happened.

**Where were the upsets in March 2018?**

Run the following command in the Query editor:

SELECT

CONCAT(school\_ncaa, " was predicted to ",IF(predicted\_label="loss","lose","win")," ",CAST(ROUND(p.prob,2)\*100 AS STRING), "% but ", IF(n.label="loss","lost","won")) AS narrative,

predicted\_label, # what the model thought

n.label, # what actually happened

ROUND(p.prob,2) AS probability,

season,

# us

seed,

school\_ncaa,

pace\_rank,

efficiency\_rank,

# them

opponent\_seed,

opponent\_school\_ncaa,

opp\_pace\_rank,

opp\_efficiency\_rank

FROM `bracketology.ncaa\_2018\_predictions` AS n,

UNNEST(predicted\_label\_probs) AS p

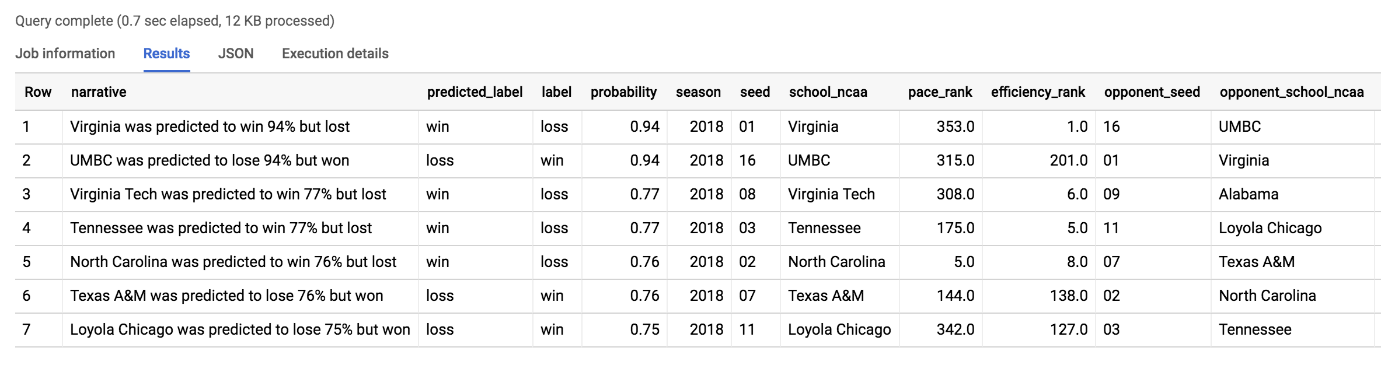
WHERE

predicted\_label <> n.label # model got it wrong

AND p.prob > .75 # by more than 75% confidence

ORDER BY prob DESC

Your outcome should look like:



The major upset was the same one our previous model found: UMBC vs Virginia. 2018 overall was a year of [huge upsets](https://medium.com/analyzing-ncaa-college-basketball-with-gcp/has-this-been-the-maddest-march-a8eb7b274d68). Will 2019 be just as wild?

**Comparing model performance**

What about where the naive model (comparing seeds) got it wrong but the advanced model got it right? Run the following command in the Query editor:

SELECT

CONCAT(opponent\_school\_ncaa, " (", opponent\_seed, ") was ",CAST(ROUND(ROUND(p.prob,2)\*100,2) AS STRING),"% predicted to upset ", school\_ncaa, " (", seed, ") and did!") AS narrative,

predicted\_label, # what the model thought

n.label, # what actually happened

ROUND(p.prob,2) AS probability,

season,

# us

seed,

school\_ncaa,

pace\_rank,

efficiency\_rank,

# them

opponent\_seed,

opponent\_school\_ncaa,

opp\_pace\_rank,

opp\_efficiency\_rank,

(CAST(opponent\_seed AS INT64) - CAST(seed AS INT64)) AS seed\_diff

FROM `bracketology.ncaa\_2018\_predictions` AS n,

UNNEST(predicted\_label\_probs) AS p

WHERE

predicted\_label = 'loss'

AND predicted\_label = n.label # model got it right

AND p.prob >= .55 # by 55%+ confidence

AND (CAST(opponent\_seed AS INT64) - CAST(seed AS INT64)) > 2 # seed difference magnitude

ORDER BY (CAST(opponent\_seed AS INT64) - CAST(seed AS INT64)) DESC

Your outcome should look like:



The model predicted a Florida St. (09) upset of Xavier (01) and they did!

The upset was correctly predicted by the new model (even when the seed ranking said otherwise) based on your new skillful features like pace and shooting efficiency. Watch the [game highlights on YouTube](https://youtu.be/xSaS22_uqPY?t=449).

**Predicting for the 2019 March Madness tournament**

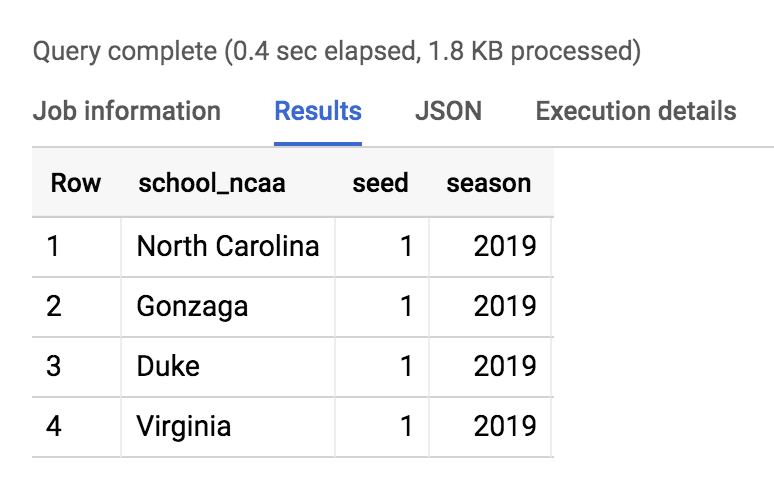
Now that we know the teams and seed rankings for March 2019, let's predict the outcome of future games.

Explore the 2019 data

Run the below query to find the top seeds

SELECT \* FROM `data-to-insights.ncaa.2019\_tournament\_seeds` WHERE seed = 1

Your outcome should look like:



Create a matrix of all possible games

Since we don't know which teams will play eachother as the tournament progresses, we'll simply have them all face eachother.

In SQL, an easy way to have a single team play every other team in a table is with a CROSS JOIN.

Run the below query to get all possible team games in the tournament.

SELECT

NULL AS label,

team.school\_ncaa AS team\_school\_ncaa,

team.seed AS team\_seed,

opp.school\_ncaa AS opp\_school\_ncaa,

opp.seed AS opp\_seed

FROM `data-to-insights.ncaa.2019\_tournament\_seeds` AS team

CROSS JOIN `data-to-insights.ncaa.2019\_tournament\_seeds` AS opp

# teams cannot play against themselves :)

WHERE team.school\_ncaa <> opp.school\_ncaa

Add in 2018 team stats (pace, efficiency)

CREATE OR REPLACE TABLE `bracketology.ncaa\_2019\_tournament` AS

WITH team\_seeds\_all\_possible\_games AS (

SELECT

NULL AS label,

team.school\_ncaa AS school\_ncaa,

team.seed AS seed,

opp.school\_ncaa AS opponent\_school\_ncaa,

opp.seed AS opponent\_seed

FROM `data-to-insights.ncaa.2019\_tournament\_seeds` AS team

CROSS JOIN `data-to-insights.ncaa.2019\_tournament\_seeds` AS opp

# teams cannot play against themselves :)

WHERE team.school\_ncaa <> opp.school\_ncaa

)

, add\_in\_2018\_season\_stats AS (

SELECT

team\_seeds\_all\_possible\_games.\*,

# bring in features from the 2018 regular season for each team

(SELECT AS STRUCT \* FROM `data-to-insights.ncaa.feature\_engineering` WHERE school\_ncaa = team AND season = 2018) AS team,

(SELECT AS STRUCT \* FROM `data-to-insights.ncaa.feature\_engineering` WHERE opponent\_school\_ncaa = team AND season = 2018) AS opp

FROM team\_seeds\_all\_possible\_games

)

# Preparing 2019 data for prediction

SELECT

label,

2019 AS season, # 2018-2019 tournament season

# our team

seed,

school\_ncaa,

# new pace metrics (basketball possession)

team.pace\_rank,

team.poss\_40min,

team.pace\_rating,

# new efficiency metrics (scoring over time)

team.efficiency\_rank,

team.pts\_100poss,

team.efficiency\_rating,

# opposing team

opponent\_seed,

opponent\_school\_ncaa,

# new pace metrics (basketball possession)

opp.pace\_rank AS opp\_pace\_rank,

opp.poss\_40min AS opp\_poss\_40min,

opp.pace\_rating AS opp\_pace\_rating,

# new efficiency metrics (scoring over time)

opp.efficiency\_rank AS opp\_efficiency\_rank,

opp.pts\_100poss AS opp\_pts\_100poss,

opp.efficiency\_rating AS opp\_efficiency\_rating,

# a little feature engineering (take the difference in stats)

# new pace metrics (basketball possession)

opp.pace\_rank - team.pace\_rank AS pace\_rank\_diff,

opp.poss\_40min - team.poss\_40min AS pace\_stat\_diff,

opp.pace\_rating - team.pace\_rating AS pace\_rating\_diff,

# new efficiency metrics (scoring over time)

opp.efficiency\_rank - team.efficiency\_rank AS eff\_rank\_diff,

opp.pts\_100poss - team.pts\_100poss AS eff\_stat\_diff,

opp.efficiency\_rating - team.efficiency\_rating AS eff\_rating\_diff

FROM add\_in\_2018\_season\_stats

Make predictions

CREATE OR REPLACE TABLE `bracketology.ncaa\_2019\_tournament\_predictions` AS

SELECT

\*

FROM

# let's predicted using the newer model

ML.PREDICT(MODEL `bracketology.ncaa\_model\_updated`, (

# let's predict on March 2019 tournament games:

SELECT \* FROM `bracketology.ncaa\_2019\_tournament`

))

Click *Check my progress* to verify the objective.

Run a queries to create tables ncaa\_2019\_tournament and ncaa\_2019\_tournament\_predictions

Check my progress

Get your predictions

SELECT

p.label AS prediction,

ROUND(p.prob,3) AS confidence,

school\_ncaa,

seed,

opponent\_school\_ncaa,

opponent\_seed

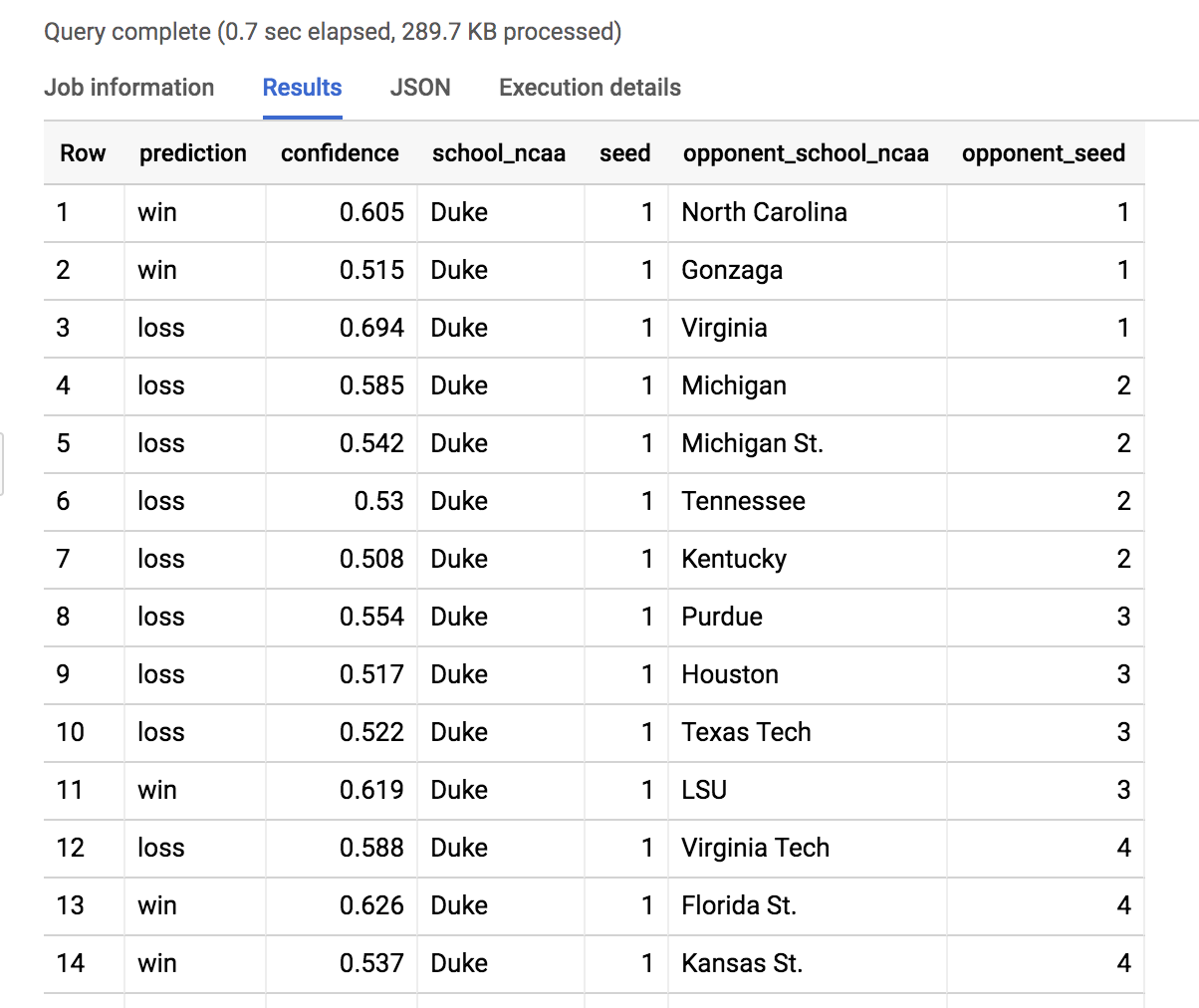
FROM `bracketology.ncaa\_2019\_tournament\_predictions`,

UNNEST(predicted\_label\_probs) AS p

WHERE p.prob >= .5

AND school\_ncaa = 'Duke'

ORDER BY seed, opponent\_seed



Here we filtered the model results to see all of Duke's possible games. Scroll to find the Duke vs North Dakota St. game.

Insight: Duke (1) is 88.5% favored to beat North Dakota St. (16) on 3/22/19.

Experiment by changing the school\_ncaa filter above to predict for the matchups in your bracket. Write down what the model confidence is and enjoy the games!

**Congratulations!**

And there you have it! You used Machine Learning to predict winning teams for the NCAA Men's basketball tournament.

Finish Your Quest

This self-paced lab is part of the Qwiklabs [NCAA® March Madness®: Bracketology with Google Cloud](https://google.qwiklabs.com/quests/58/) and [BigQuery for BigQuery for Machine Learning](https://google.qwiklabs.com/quests/71) Quests. A Quest is a series of related labs that form a learning path. Completing a Quest earns you a badge to recognize your achievement. You can make your badge (or badges) public and link to them in your online resume or social media account. Enroll in either Quest and get immediate completion credit if you've taken this lab. See other available Qwiklabs Quests](http://google.qwiklabs.com/catalog).

Take your next lab

We hope you enjoyed learning all about big data exploration, and how quickly you can create machine learning models inside of BigQuery. Try these out next:

* [Exploring NCAA Data with BigQuery](https://google.qwiklabs.com/focuses/984?parent=catalog)
* [Introduction to SQL for BigQuery and Cloud SQL](https://google.qwiklabs.com/focuses/2801?parent=catalog)

Next steps / learn more

* Want to learn more about basketball metrics and analysis? Check out [additional analysis](https://medium.com/analyzing-ncaa-college-basketball-with-gcp) from team behind the Google Cloud NCAA tournament ads and predictions.
* See what you can do with ML with [Using Machine Learning on Compute Engine to Make Product Recommendations](https://cloud.google.com/solutions/recommendations-using-machine-learning-on-compute-engine)
* [Data Science quest](https://www.qwiklabs.com/quests/28?locale=en)

Google Cloud Training & Certification

...helps you make the most of Google Cloud technologies. [Our classes](https://cloud.google.com/training/courses) include technical skills and best practices to help you get up to speed quickly and continue your learning journey. We offer fundamental to advanced level training, with on-demand, live, and virtual options to suit your busy schedule. [Certifications](https://cloud.google.com/certification/) help you validate and prove your skill and expertise in Google Cloud technologies.

Manual Last Updated August 21, 2020

Lab Last Tested August 21, 2020

Copyright 2020 Google LLC All rights reserved. Google and the Google logo are trademarks of Google LLC. All other company and product names may be trademarks of the respective companies with which they are associated.