Bracketology

Bracketology is the process of predicting the field of college basketball participants in the NCAA Basketball Tournament, named as such because it is commonly used to fill in tournament brackets for the postseason. It incorporates some method of predicting what the NCAA Selection Committee will use as its Ratings Percentage Index in order to determine at-large (non-conference winning) teams to complete the field of 64 teams, and, to seed the field by ranking all teams from first through sixty-eighth. Bracketology also encompasses the process of predicting the winners of each of the brackets. In recent years the concept of bracketology has been applied to areas other than basketball.

Various methods are used to predict the winners in a bracket. While some use math and statistics, others make selections based on team mascots or colors. President Barack Obama became famous for his bracket predictions. After entering office, he presented his projected winners annually on ESPN in a segment called Barack-etology.

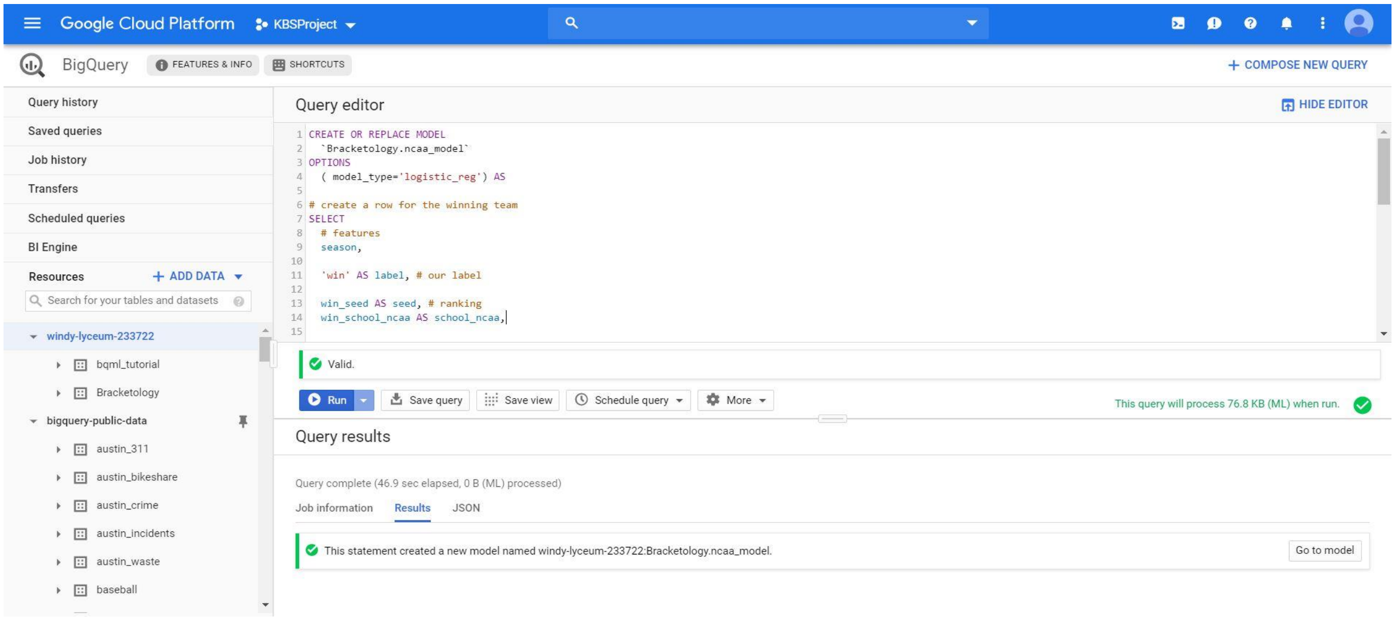
Bracketology as a discipline has spread beyond a focus on basketball, into other sports, as well as pop culture, history, nature, and other topics where a loose application of binary opposition may be profitable for study or enjoyment, albeit without the label of "bracketology" itself.

Below are the steps of the Bracketology for NCAA Men’s Basketball.

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

Choosing a model type

We have chosen 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 is that the ML model will do all the math and optimization during model training.



```
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,

```

Valid.

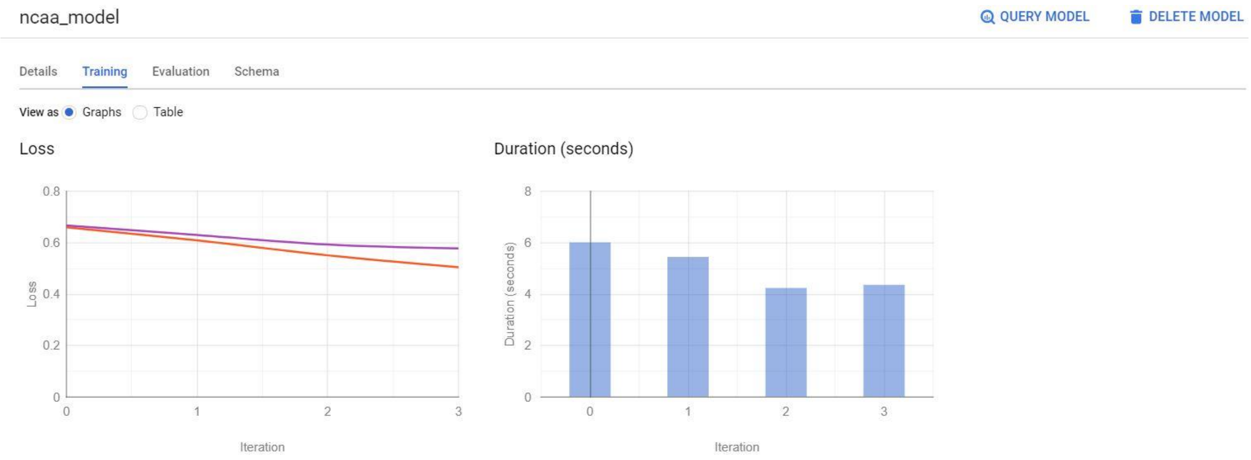
This query will process 76.8 KB (ML) when run.

Query complete (46.9 sec elapsed, 0 B (ML) processed)

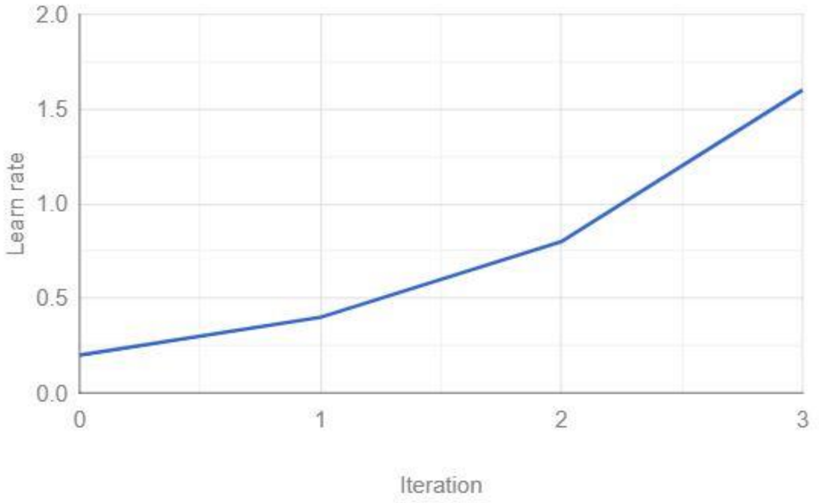
This statement created a new model named windy-lyceum-233722:Bracketology.ncaa_model.

Step 2: -

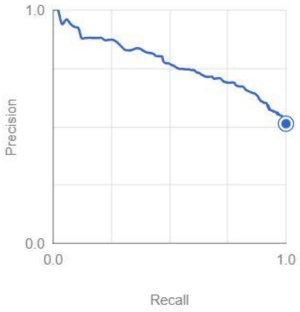
View model training details and stats.



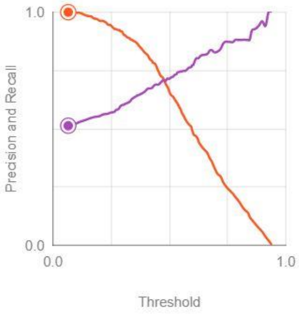
Learn rate



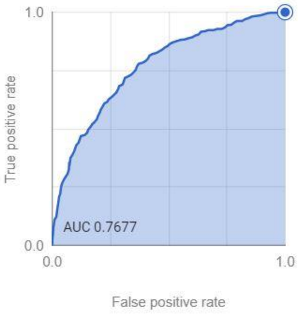
Precision-Recall curve

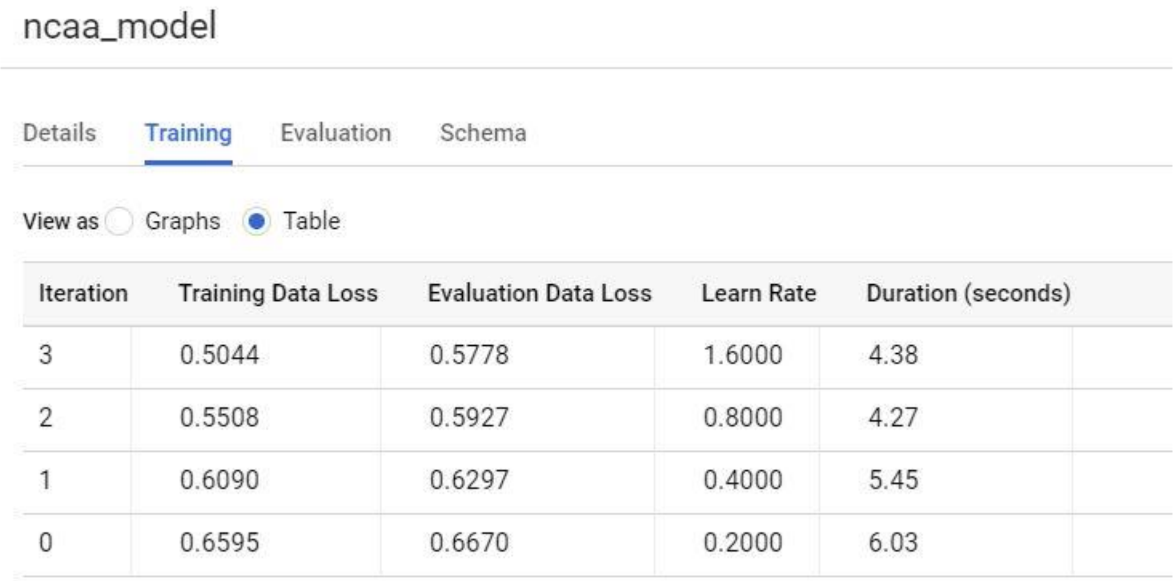


Precision and Recall vs Threshold



ROC curve





ncaa_model

View as ○ Graphs ● Table

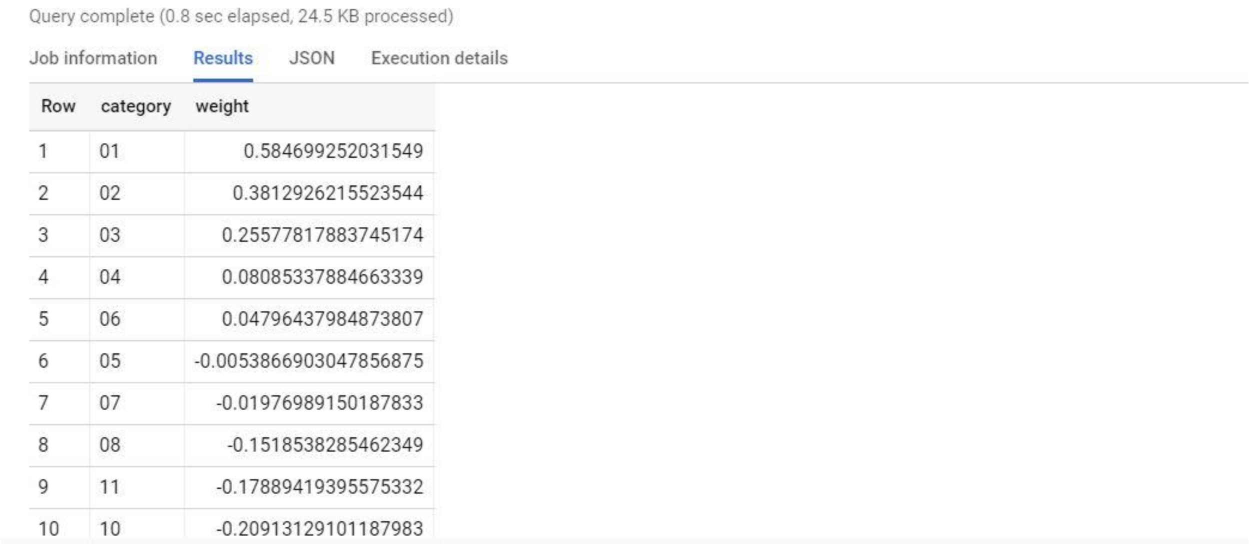
Iteration	Training Data Loss	Evaluation Data Loss	Learn Rate	Duration (seconds)
3	0.5044	0.5778	1.6000	4.38
2	0.5508	0.5927	0.8000	4.27
1	0.6090	0.6297	0.4000	5.45
0	0.6595	0.6670	0.2000	6.03

Step 3: -

See what the model learned about our features.

To learn this, we can execute the following query in bigquery.

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



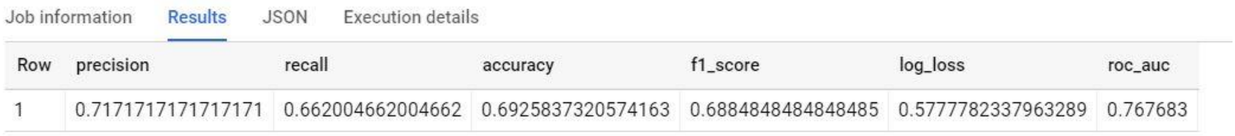
Query complete (0.8 sec elapsed, 24.5 KB processed)

Row	category	weight
1	01	0.584699252031549
2	02	0.3812926215523544
3	03	0.25577817883745174
4	04	0.08085337884663339
5	06	0.04796437984873807
6	05	-0.0053866903047856875
7	07	-0.01976989150187833
8	08	-0.1518538285462349
9	11	-0.17889419395575332
10	10	-0.20913129101187983

As we can see, if the seed 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.

Step 4: -

Evaluate model performance


Job information
Results
JSON
Execution details

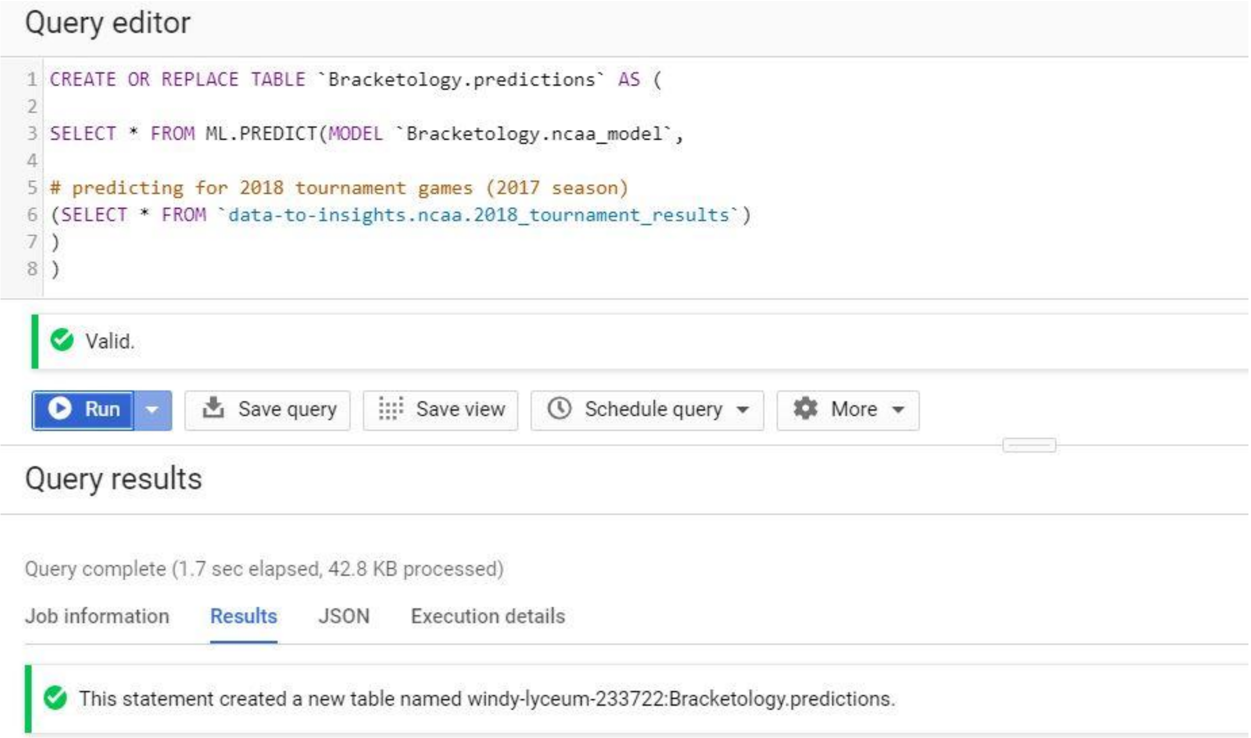
Row	precision	recall	accuracy	f1_score	log_loss	roc_auc
1	0.7171717171717171	0.662004662004662	0.6925837320574163	0.6884848484848485	0.5777782337963289	0.767683

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

Step 5: -

Making predictions

We can make predictions is as simple as calling ML.PREDICT on a trained model and passing through the dataset we want to predict on.


Query editor

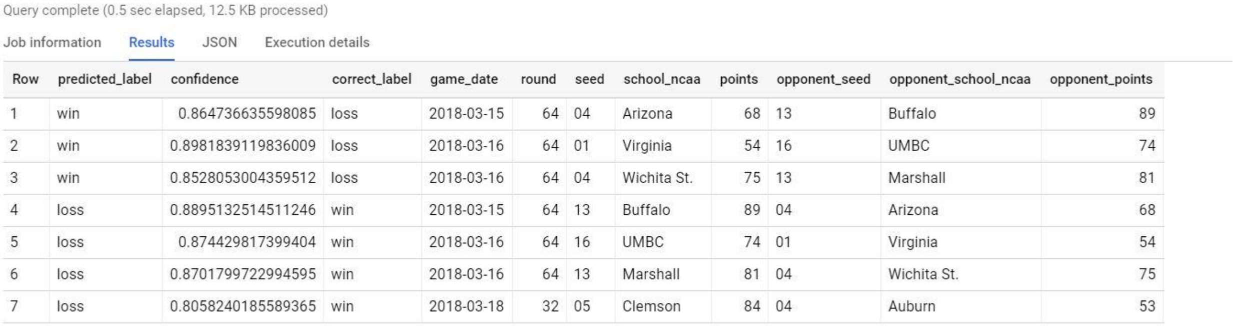
```
CREATE OR REPLACE TABLE `Bracketology.predictions` AS (

SELECT * FROM ML.PREDICT(MODEL `Bracketology.ncaa_model`,

# predicting for 2018 tournament games (2017 season)
(SELECT * FROM `data-to-insights.ncaa.2018_tournament_results`)
)
)
```

✓ Valid.

Query results

Query complete (1.7 sec elapsed, 42.8 KB processed)

✓ This statement created a new table named windy-lyceum-233722:Bracketology.predictions.



Query complete (0.5 sec elapsed, 12.5 KB processed)

Row	predicted_label	confidence	correct_label	game_date	round	seed	school_ncaa	points	opponent_seed	opponent_school_ncaa	opponent_points
1	win	0.864736635598085	loss	2018-03-15	64	04	Arizona	68	13	Buffalo	89
2	win	0.8981839119836009	loss	2018-03-16	64	01	Virginia	54	16	UMBC	74
3	win	0.8528053004359512	loss	2018-03-16	64	04	Wichita St.	75	13	Marshall	81
4	loss	0.8895132514511246	win	2018-03-15	64	13	Buffalo	89	04	Arizona	68
5	loss	0.874429817399404	win	2018-03-16	64	16	UMBC	74	01	Virginia	54
6	loss	0.8701799722994595	win	2018-03-16	64	13	Marshall	81	04	Wichita St.	75
7	loss	0.8058240185589365	win	2018-03-18	32	05	Clemson	84	04	Auburn	53

Step 6: -

Now, we can check how our model performed on predictions.

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

SELECT * FROM `bracketology.predictions`

WHERE predicted_label <> label

Out of 134 predictions (67 March tournament games), our model got it wrong 38 times.

70% overall for the 2018 tournament matchup.

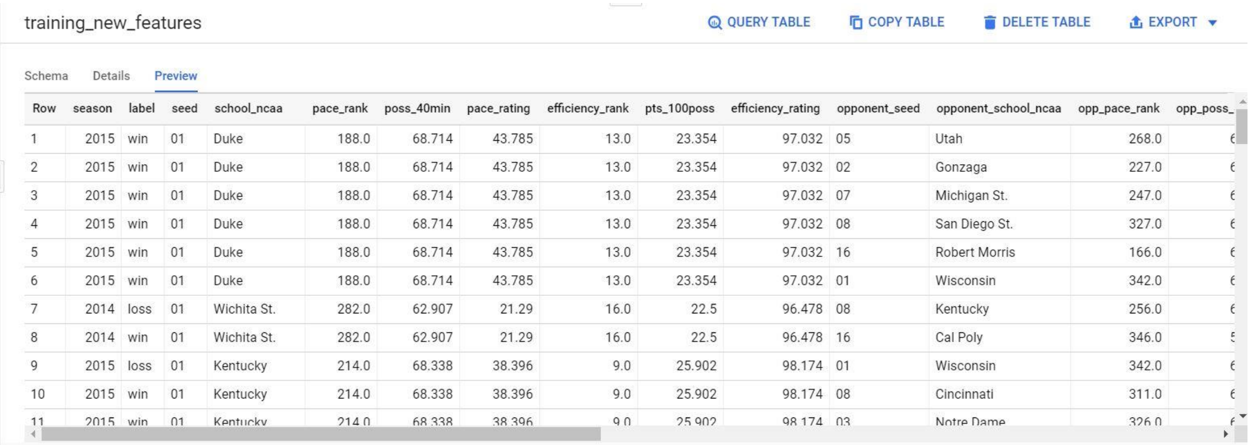
Part 2: Using skillful ML model features

Step 1: -

Create a new ML dataset with these skillful features. Below are some of the features

season, win, win_seed, , win_school_ncaa, lose_seed, lose_school_ncaa

Now, we will preview the new features

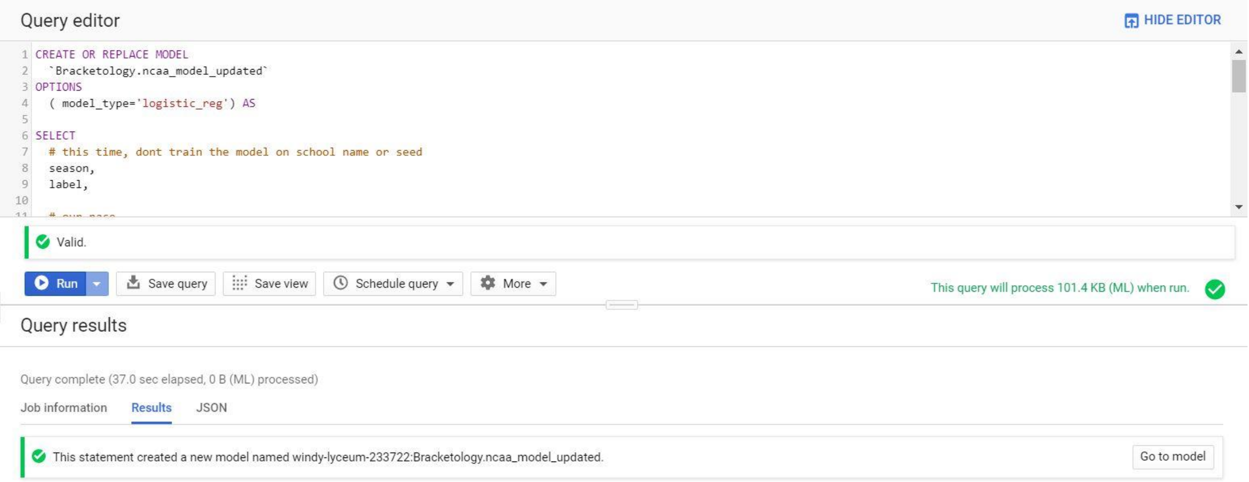


training_new_features

Row	season	label	seed	school_ncaa	pace_rank	poss_40min	pace_rating	efficiency_rank	pts_100poss	efficiency_rating	opponent_seed	opponent_school_ncaa	opp_pace_rank	opp_poss_
1	2015	win	01	Duke	188.0	68.714	43.785	13.0	23.354	97.032	05	Utah	268.0	6
2	2015	win	01	Duke	188.0	68.714	43.785	13.0	23.354	97.032	02	Gonzaga	227.0	6
3	2015	win	01	Duke	188.0	68.714	43.785	13.0	23.354	97.032	07	Michigan St.	247.0	6
4	2015	win	01	Duke	188.0	68.714	43.785	13.0	23.354	97.032	08	San Diego St.	327.0	6
5	2015	win	01	Duke	188.0	68.714	43.785	13.0	23.354	97.032	16	Robert Morris	166.0	6
6	2015	win	01	Duke	188.0	68.714	43.785	13.0	23.354	97.032	01	Wisconsin	342.0	6
7	2014	loss	01	Wichita St.	282.0	62.907	21.29	16.0	22.5	96.478	08	Kentucky	256.0	6
8	2014	win	01	Wichita St.	282.0	62.907	21.29	16.0	22.5	96.478	16	Cal Poly	346.0	5
9	2015	loss	01	Kentucky	214.0	68.338	38.396	9.0	25.902	98.174	01	Wisconsin	342.0	6
10	2015	win	01	Kentucky	214.0	68.338	38.396	9.0	25.902	98.174	08	Cincinnati	311.0	6
11	2015	win	01	Kentucky	214.0	68.338	38.396	9.0	25.902	98.174	03	Notre Dame	326.0	6

Step 2: -

Train the new model.



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,

  # own pace
```

Valid.

This query will process 101.4 KB (ML) when run.

Query results

Query complete (37.0 sec elapsed, 0 B (ML) processed)

This statement created a new model named windy-lyceum-233722:Bracketology.ncaa_model_updated.

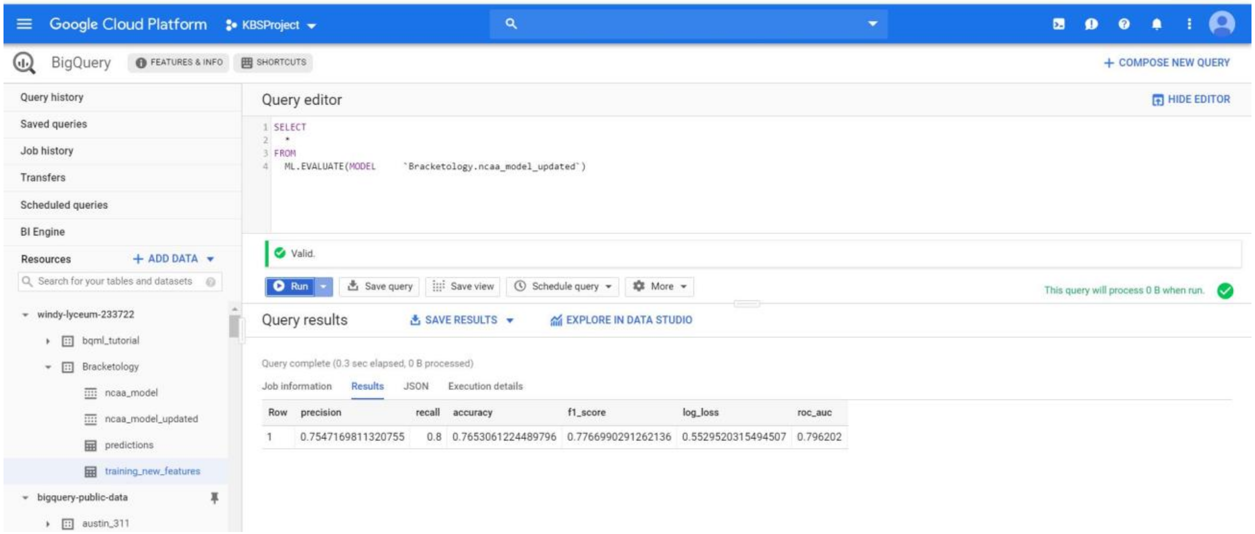
Step 3: -

Evaluate the new model's performance.

SELECT * FROM ML.EVALUATE(MODEL `bracketology.ncaa_model_updated`)

Step 4: -

Inspect what the model learned.



```
SELECT
  *
FROM
  ML.EVALUATE(MODEL     `Bracketology.ncaa_model_updated`)
```

Valid.

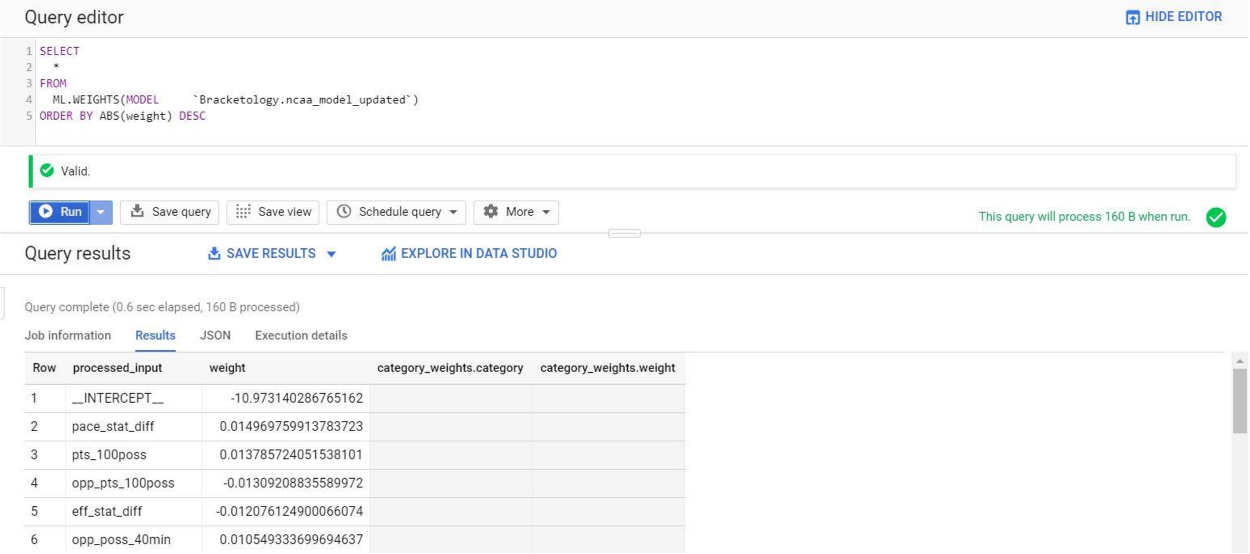
This query will process 0 B when run.

Query complete (0.3 sec elapsed, 0 B processed)

Row	precision	recall	accuracy	f1_score	log_loss	roc_auc
1	0.7547169811320755	0.8	0.7653061224489796	0.7766990291262136	0.5529520315494507	0.796202

Step 5: -

Now, we will inspect what the model has learned. Which features does the model weigh the most in win / loss outcome? Here, 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.



```
SELECT
  *
FROM
  ML.WEIGHTS(MODEL     `Bracketology.ncaa_model_updated`)
ORDER BY ABS(weight) DESC
```

Valid.

This query will process 160 B when run.

Query complete (0.6 sec elapsed, 160 B processed)

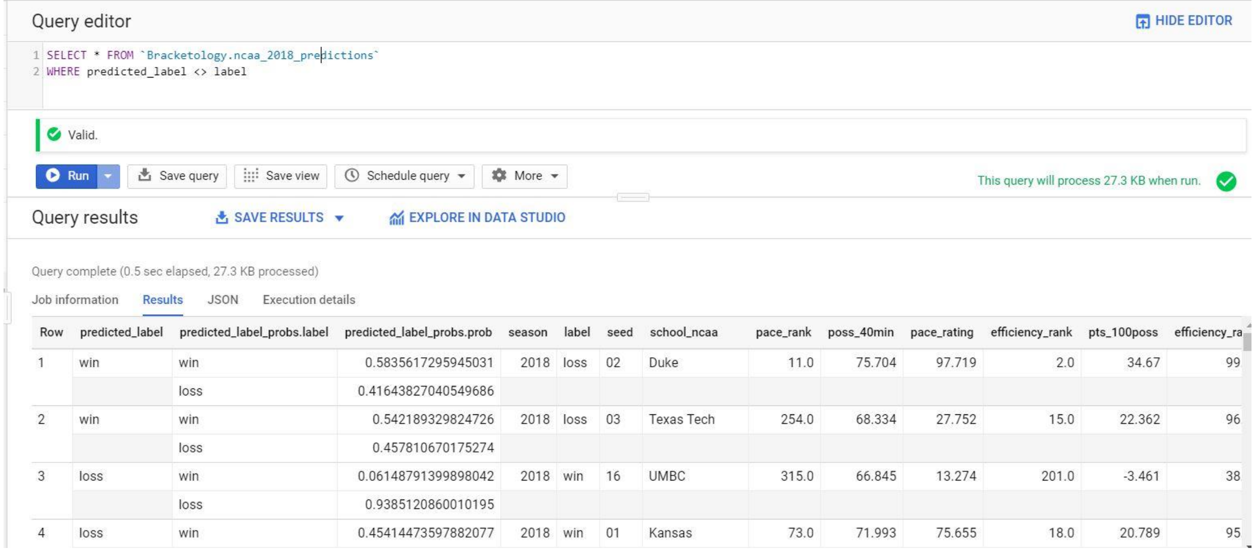
Row	processed_input	weight	category_weights.category	category_weights.weight
1	__INTERCEPT__	-10.973140286765162		
2	pace_stat_diff	0.014969759913783723		
3	pts_100poss	0.013785724051538101		
4	opp_pts_100poss	-0.01309208835589972		
5	eff_stat_diff	-0.012076124900066074		
6	opp_poss_40min	0.010549333699694637		

Step 6: -

Now, we will make prediction. For that we will use 2018 season.

Step 7: -

Prediction analysis: -



```
SELECT * FROM `Bracketology.ncaa_2018_predictions`
WHERE predicted_label <> label
```

Valid.

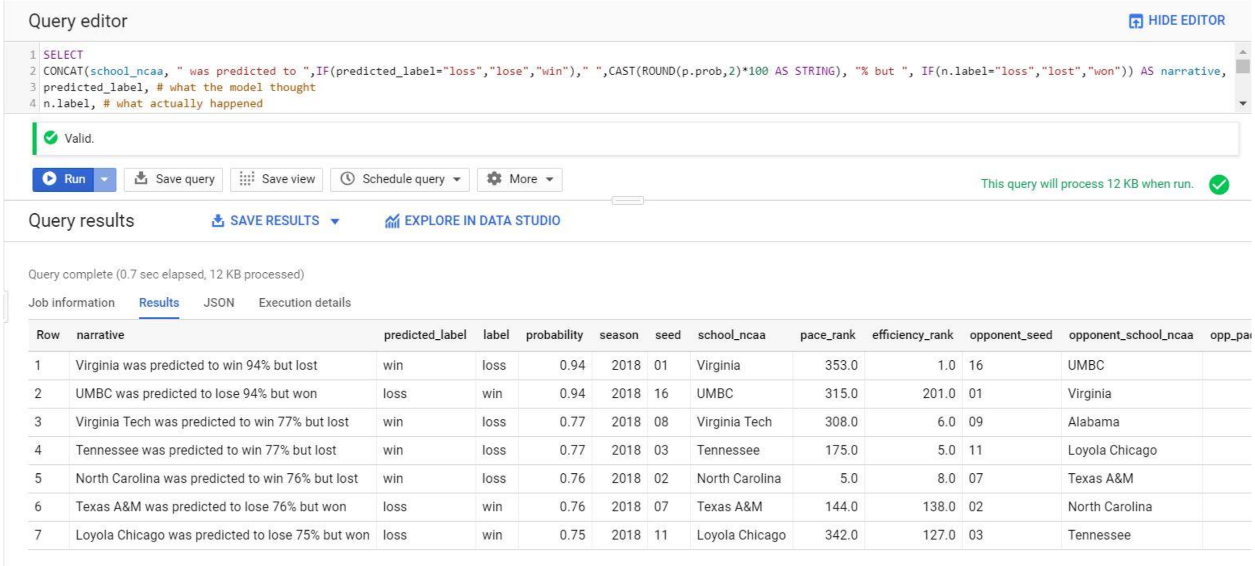
This query will process 27.3 KB when run.

Query complete (0.5 sec elapsed, 27.3 KB processed)

Row	predicted_label	predicted_label_probs.label	predicted_label_probs.prob	season	label	seed	school_ncaa	pace_rank	poss_40min	pace_rating	efficiency_rank	pts_100poss	efficiency_ra
1	win	win	0.5835617295945031	2018	loss	02	Duke	11.0	75.704	97.719	2.0	34.67	99
		loss	0.41643827040549686										
2	win	win	0.542189329824726	2018	loss	03	Texas Tech	254.0	68.334	27.752	15.0	22.362	96
		loss	0.457810670175274										
3	loss	win	0.06148791399898042	2018	win	16	UMBC	315.0	66.845	13.274	201.0	-3.461	38
		loss	0.9385120860010195										
4	loss	win	0.45414473597882077	2018	win	01	Kansas	73.0	71.993	75.655	18.0	20.789	95

Step 8: -

Where were the upsets in March 2018?



```
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
```

Valid.

This query will process 12 KB when run.

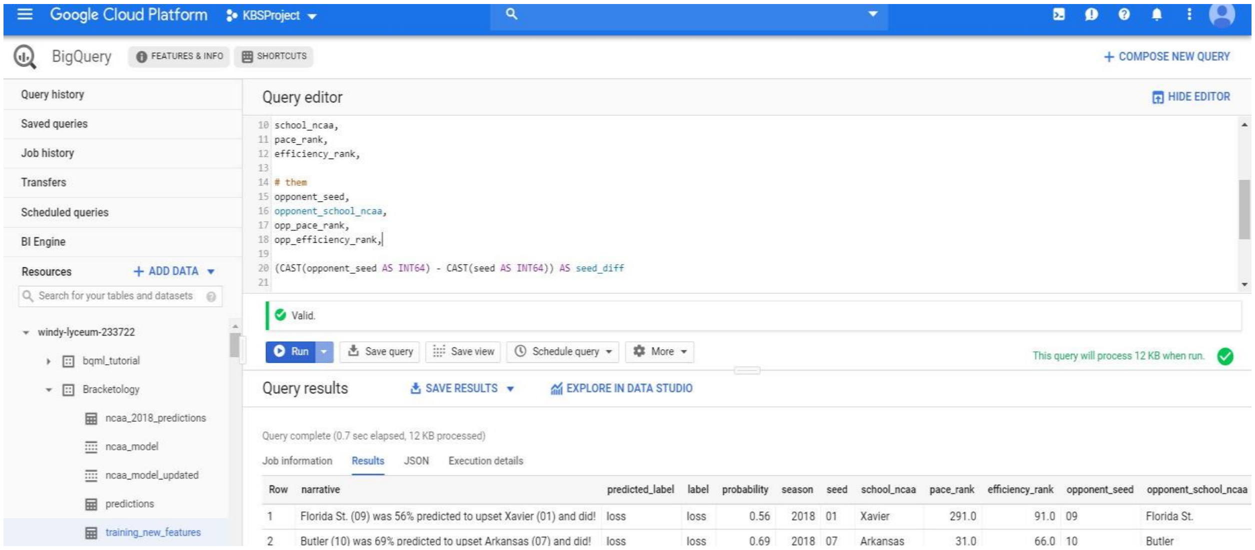
Query complete (0.7 sec elapsed, 12 KB processed)

Row	narrative	predicted_label	label	probability	season	seed	school_ncaa	pace_rank	efficiency_rank	opponent_seed	opponent_school_ncaa	opp_pa
1	Virginia was predicted to win 94% but lost	win	loss	0.94	2018	01	Virginia	353.0	1.0	16	UMBC	
2	UMBC was predicted to lose 94% but won	loss	win	0.94	2018	16	UMBC	315.0	201.0	01	Virginia	
3	Virginia Tech was predicted to win 77% but lost	win	loss	0.77	2018	08	Virginia Tech	308.0	6.0	09	Alabama	
4	Tennessee was predicted to win 77% but lost	win	loss	0.77	2018	03	Tennessee	175.0	5.0	11	Loyola Chicago	
5	North Carolina was predicted to win 76% but lost	win	loss	0.76	2018	02	North Carolina	5.0	8.0	07	Texas A&M	
6	Texas A&M was predicted to lose 76% but won	loss	win	0.76	2018	07	Texas A&M	144.0	138.0	02	North Carolina	
7	Loyola Chicago was predicted to lose 75% but won	loss	win	0.75	2018	11	Loyola Chicago	342.0	127.0	03	Tennessee	

The major upset was the same which we have found in the previous model : UMBC vs Virginia.

Step 9: -

Comparing model performance.



```
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

```

Valid.

This query will process 12 KB when run.

Query complete (0.7 sec elapsed, 12 KB processed)

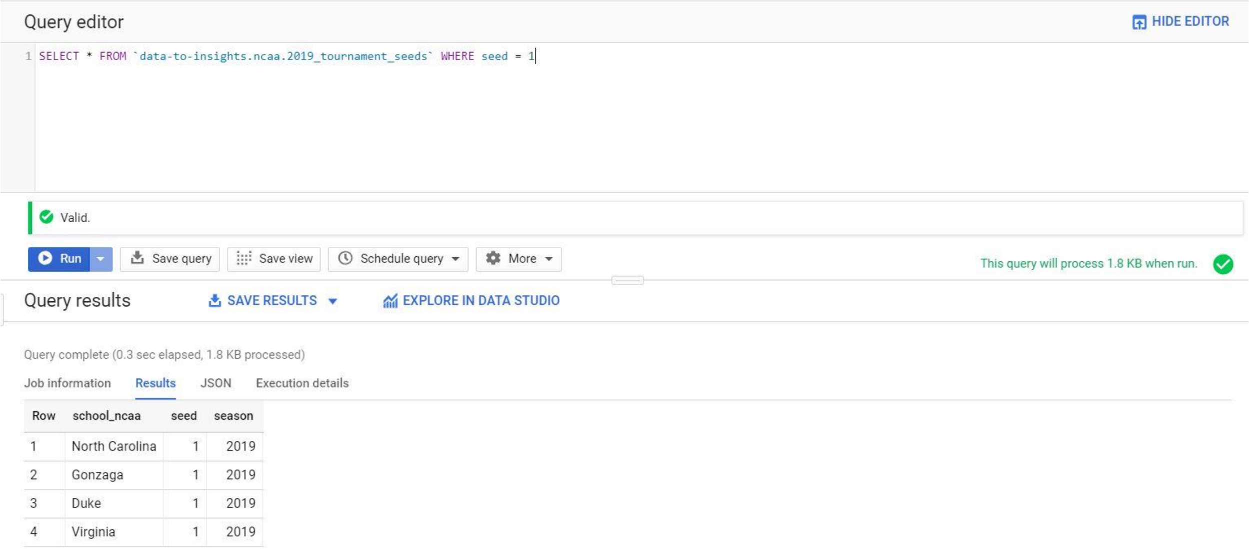
Row	narrative	predicted_label	label	probability	season	seed	school_ncaa	pace_rank	efficiency_rank	opponent_seed	opponent_school_ncaa
1	Florida St. (09) was 56% predicted to upset Xavier (01) and did!	loss	loss	0.56	2018	01	Xavier	291.0	91.0	09	Florida St.
2	Butler (10) was 69% predicted to upset Arkansas (07) and did!	loss	loss	0.69	2018	07	Arkansas	31.0	66.0	10	Butler

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 new skillful features like pace and shooting efficiency.

Step 10: -

Predicting for the 2019 March Madness tournament.



```
SELECT * FROM `data-to-insights.ncaa.2019_tournament_seeds` WHERE seed = 1
```

Valid.

This query will process 1.8 KB when run.

Query complete (0.3 sec elapsed, 1.8 KB processed)

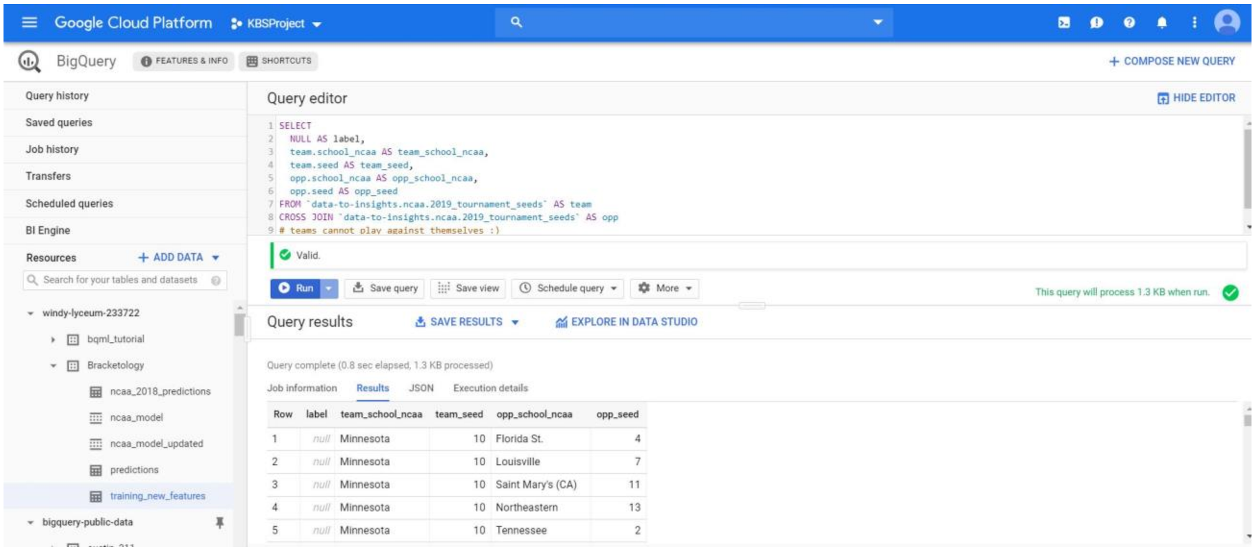
Row	school_ncaa	seed	season
1	North Carolina	1	2019
2	Gonzaga	1	2019
3	Duke	1	2019
4	Virginia	1	2019

Step 11: -

Create a matrix of all possible games.

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

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



```
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 :)
```

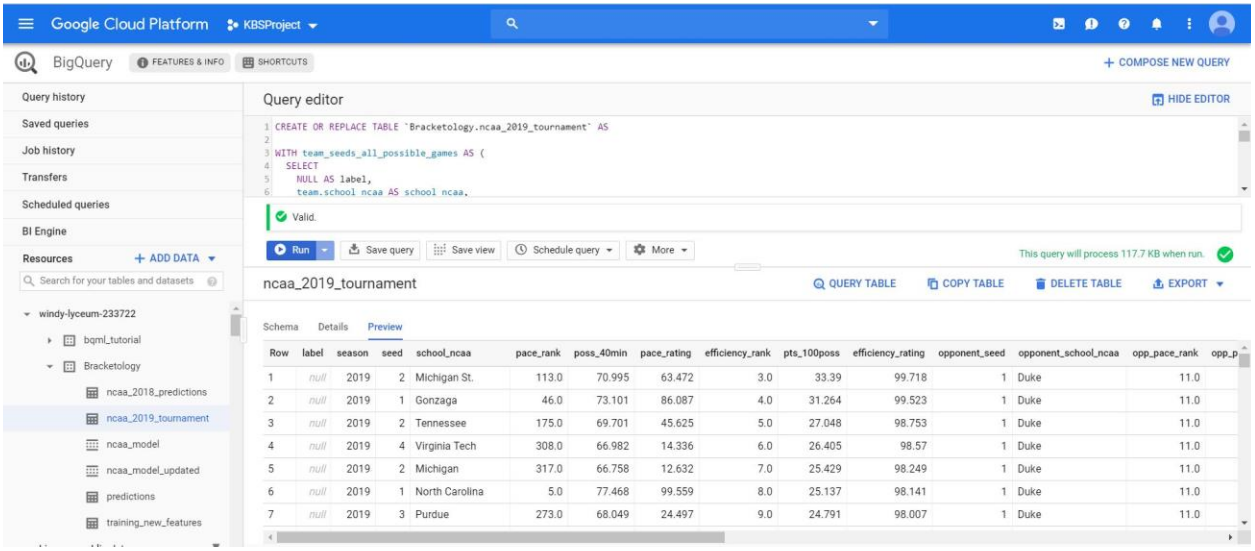
Valid.

This query will process 1.3 KB when run.

Query complete (0.8 sec elapsed, 1.3 KB processed)

Row	label	team_school_ncaa	team_seed	opp_school_ncaa	opp_seed
1	null	Minnesota	10	Florida St.	4
2	null	Minnesota	10	Louisville	7
3	null	Minnesota	10	Saint Mary's (CA)	11
4	null	Minnesota	10	Northeastern	13
5	null	Minnesota	10	Tennessee	2

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,
```

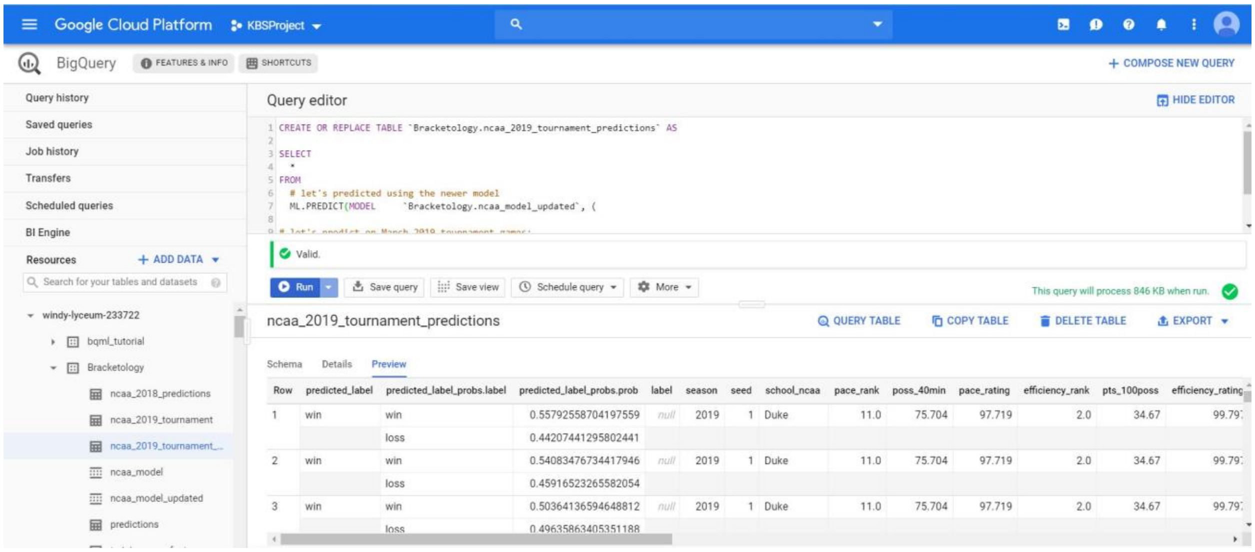
Valid.

This query will process 117.7 KB when run.

ncaa_2019_tournament

Row	label	season	seed	school_ncaa	pace_rank	poss_40min	pace_rating	efficiency_rank	pts_100poss	efficiency_rating	opponent_seed	opponent_school_ncaa	opp_pace_rank	opp_p
1	null	2019	2	Michigan St.	113.0	70.995	63.472	3.0	33.39	99.718	1	Duke	11.0	
2	null	2019	1	Gonzaga	46.0	73.101	86.087	4.0	31.264	99.523	1	Duke	11.0	
3	null	2019	2	Tennessee	175.0	69.701	45.625	5.0	27.048	98.753	1	Duke	11.0	
4	null	2019	4	Virginia Tech	308.0	66.982	14.336	6.0	26.405	98.57	1	Duke	11.0	
5	null	2019	2	Michigan	317.0	66.758	12.632	7.0	25.429	98.249	1	Duke	11.0	
6	null	2019	1	North Carolina	5.0	77.468	99.559	8.0	25.137	98.141	1	Duke	11.0	
7	null	2019	3	Purdue	273.0	68.049	24.497	9.0	24.791	98.007	1	Duke	11.0	

Prepare 2019 data for prediction.



```
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:
```

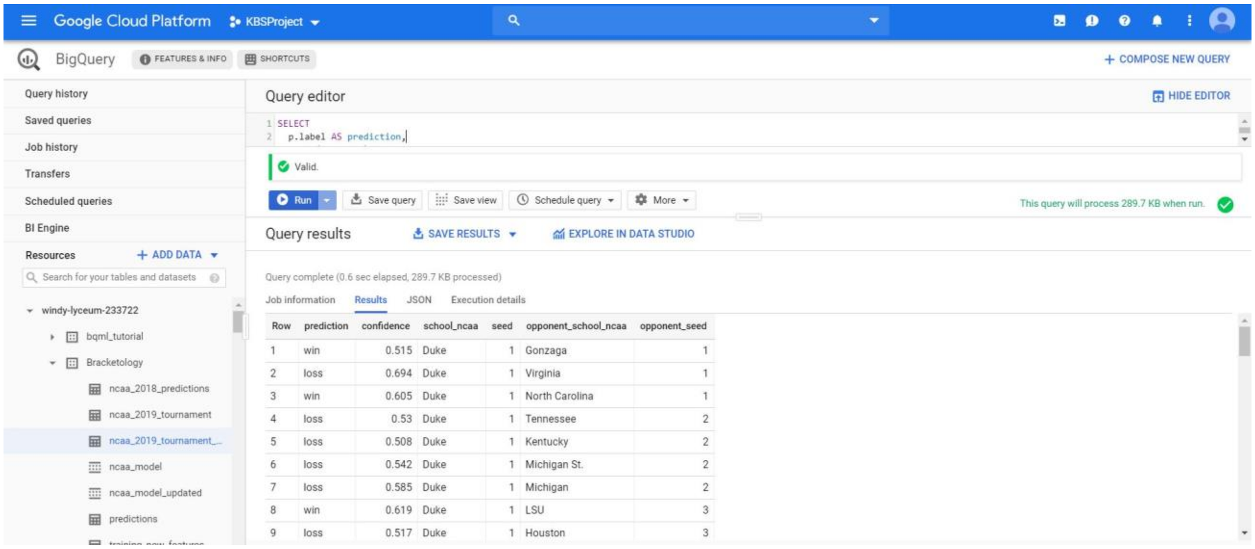
Valid.

This query will process 846 KB when run.

ncaa_2019_tournament_predictions

Row	predicted_label	predicted_label_probs.label	predicted_label_probs.prob	label	season	seed	school_ncaa	pace_rank	poss_40min	pace_rating	efficiency_rank	pts_100poss	efficiency_rating
1	win	win	0.55792558704197559	null	2019	1	Duke	11.0	75.704	97.719	2.0	34.67	99.79
		loss	0.44207441295802441										
2	win	win	0.54083476734417946	null	2019	1	Duke	11.0	75.704	97.719	2.0	34.67	99.79
		loss	0.45916523265582054										
3	win	win	0.50364136594648812	null	2019	1	Duke	11.0	75.704	97.719	2.0	34.67	99.79
		loss	0.49635863405351188										

Make predictions



```
SELECT
  p.label AS prediction,
```

Valid.

This query will process 289.7 KB when run.

Query complete (0.6 sec elapsed, 289.7 KB processed)

Row	prediction	confidence	school_ncaa	seed	opponent_school_ncaa	opponent_seed
1	win	0.515	Duke	1	Gonzaga	1
2	loss	0.694	Duke	1	Virginia	1
3	win	0.605	Duke	1	North Carolina	1
4	loss	0.53	Duke	1	Tennessee	2
5	loss	0.508	Duke	1	Kentucky	2
6	loss	0.542	Duke	1	Michigan St.	2
7	loss	0.585	Duke	1	Michigan	2
8	win	0.619	Duke	1	LSU	3
9	loss	0.517	Duke	1	Houston	3

Here we filtered the model results to see all of Duke's possible games and we can see that model has predicted that Virginia has the highest chance of winning and that is what happened. Virginia won 2019 NCAA Basketball men’s final.