BigQuery Machine Learning

The data used in this lab comes from the following sources:

- Pappalardo et al., (2019) **A public data set of spatio-temporal match events in soccer competitions**, Nature Scientific Data 6:236, https://www.nature.com/articles/s41597-019-0247-7
- Pappalardo et al. (2019) **PlayerRank: Data-driven Performance Evaluation and Player Ranking in Soccer via a Machine Learning Approach**. ACM Transactions on Intelligent Systems and Technologies (TIST) 10, 5, Article 59 (September 2019), 27 pages. DOI: https://doi.org/10.1145/3343172

In this lab, you will:

- Write functions in BigQuery to help with calculations to be performed on soccer shot data
- Create and evaluate "expected goals" models using BigQuery ML
- Apply an expected goals model to "new" data using BigQuery ML

Calculate shot distance and shot angle

In this section, you will create some user-defined functions in BigQuery that help with shot distance and angle calculations, helping to prepare the soccer event data for eventual use in an ML model.

Calculate shot distance from (x,y) coordinates

To motivate the need for the function, start by looking at the **positions** field in the **events** table a bit more - this is a repeated field that contains 1 or more (x, y) pairs per event. Per Wyscout, a leading data company in the soccer industry that provided this data, these represent the origin and (if applicable) destination positions associated with the event, on a 0-100 scale representing the percentage of the field from the perspective of the attacking team. See the screenshot of the table below

showing positions corresponding to a few different event types for a few example events.

eventName	playerId	subEventName	id	positions.x	positions.y
Foul	83575	Late card foul	88179369	30	60
Free Kick	14922	Free kick shot	88520527	72	62
				100	100
Goalkeeper leaving line	83574	Goalkeeper leaving line	88743724	0	0
				24	93
Pass	78488	Hand pass	88223367	14	57
				17	34
Save attempt	78488	Reflexes	88223382	100	100
				3	58

The first (x, y) pair in positions for events that are shots represent the location on the field from which the shot originated.

Calculating the distance of a shot from a given location to the goal involves using the midpoint of the goal mouth (100, 50) as the ending location, the known average dimensions of a soccer field (105x68, per Wikipedia; there is no standard field size) and the 2-dimensional distance formula.

1. In the query **Editor**, copy and paste the following code:

```
CREATE FUNCTION `soccer.GetShotDistanceToGoal`(x INT64, y INT64)
RETURNS FLOAT64
AS (
 /* Translate 0-100 (x,y) coordinate-based distances to absolute positions
 using "average" field dimensions of 105x68 before combining in 2D dist calc */
 SQRT(
   POW((100 - x) * 105/100, 2) +
   POW((50 - y) * 68/100, 2)
   )
 );
```

This specifies the name of the function, the inputs and their types (2 integers), the output type (a float), and the actual logic implementing the 2-dimensional distance formula on the scaled x- and y-coordinate distances.

2. Click **Run**. The **Query results** section should display a message saying This statement created a new function named: **<PROJECT_NAME>.soccer.GetShotDistanceToGoal.**

In this section you used BigQuery to make functions to calculate shot distance and angle. In the next section learn how to build a model to calculate expected goals using the distance and angle of each shot.

Create expected goals models using BigQuery ML

In this section, you will use BigQuery ML to create and execute machine learning models in BigQuery using standard SQL queries. In this case, you will build expected goals models on the soccer event data to predict the likelihood of a shot going in for a goal given its type, distance, and angle (these are known to be good predictors of the chance of a goal, as demonstrated in the BigQuery Soccer Data Analytical Insight lab).

Expected goals models are commonly used in soccer analytics to measure the quality of shots and finishing/saving ability given shot quality, and they have a variety of applications in both retrospective match analysis and making projections.

Fitting a logistic regression model for expected goals

First, you will fit a logistic regression model to the data.

1. In the query **Editor**, click **COMPOSE NEW QUERY**.
2. Add the following code into the query **Editor**.

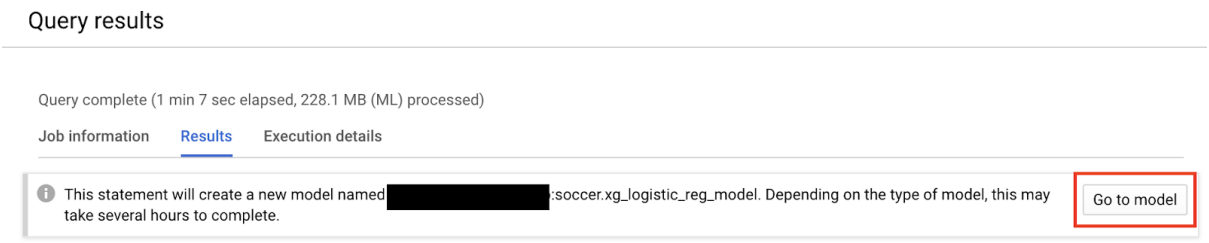
```
CREATE MODEL `soccer.xg_logistic_reg_model`
OPTIONS(
 model_type = 'LOGISTIC_REG',
 input_label_cols = ['isGoal']
 ) AS
SELECT
 Events.subEventName AS shotType,
 /* 101 is known Tag for 'goals' from goals table */
 (101 IN UNNEST(Events.tags.id)) AS isGoal,
```

```
  `soccer.GetShotDistanceToGoal`(Events.positions[ORDINAL(1)].x,
    Events.positions[ORDINAL(1)].y) AS shotDistance,
 `soccer.GetShotAngleToGoal`(Events.positions[ORDINAL(1)].x,
    Events.positions[ORDINAL(1)].y) AS shotAngle
FROM
 `soccer.events` Events
LEFT JOIN
 `soccer.matches` Matches ON
    Events.matchId = Matches.wyId
LEFT JOIN
 `soccer.competitions` Competitions ON
    Matches.competitionId = Competitions.wyId
WHERE
 /* Filter out World Cup matches for model fitting purposes */
 Competitions.name != 'World Cup' AND
 /* Includes both "open play" & free kick shots (including penalties) */
 (
    eventName = 'Shot' OR
    (eventName = 'Free Kick' AND subEventName IN ('Free kick shot', 'Penalty'))
 )
;
```

The top section is the actual model generation code, specifying the type of model and label for the outcome variable. The SELECT statement creates the **isGoal** outcome variable along with features of interest from the event data, including the shot distance and angle calculated using the functions created in the previous step. The joins enable the determination of which competition each shot came from, which is employed to filter out the World Cup data from this modeling (saved for use later on after the model is fit).

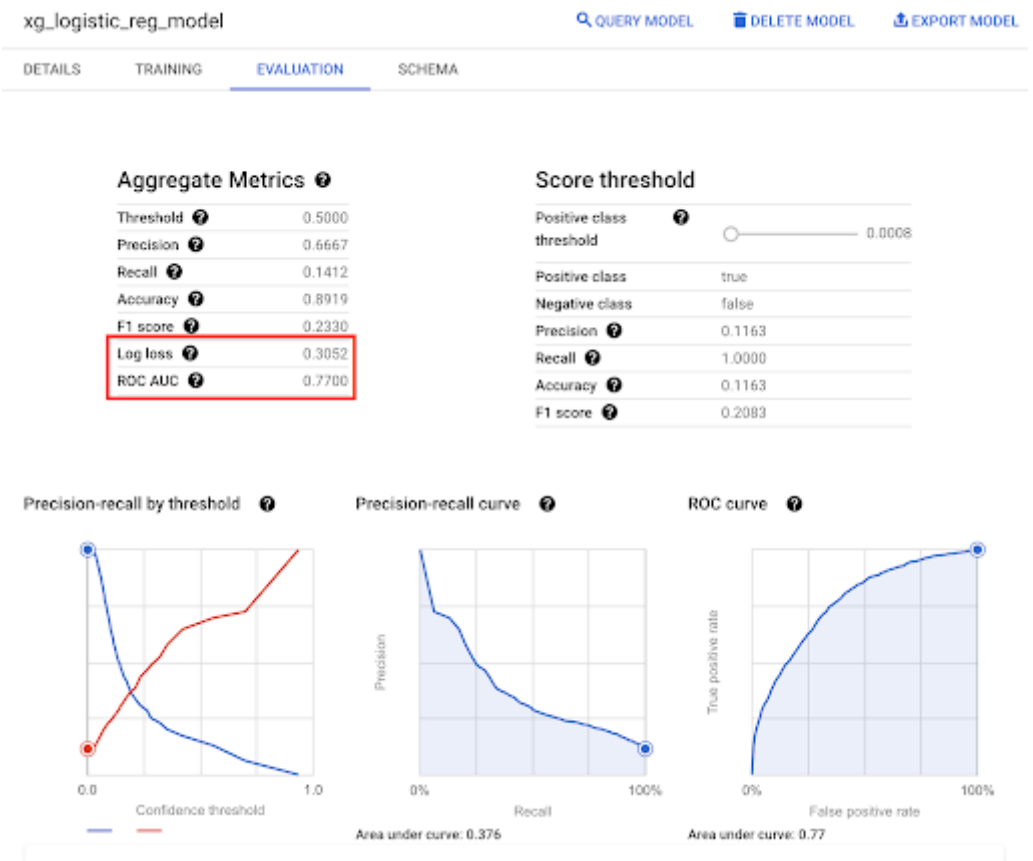
3. Click **Run**.
4. The **Query results** section should display a message saying This statement will create a new model named: **<PROJECT_NAME>:soccer.xg_logistic_reg_model**. Depending on the type of model, this may take several hours to complete. In this case, it should only take a minute or two to finish.

5. Once the model is done training - look for a "Query complete" notification in the **Query results** section - click on **Go to model** at the far right next to the message about model creation.



This will open up a new tab that has information about the model that was just trained. Click to the **EVALUATION** tab and take a look at some of the metrics, particularly **"Log loss"** and **"ROC AUC"** under **Aggregate Metrics**.

***NOTE**: Results may vary slightly based on the inherent randomness in the model fitting procedure.*



*Since the primary interest lies in the accuracy of goal probabilities from the model (not explicitly accuracy in terms of predicting 0s and 1s at a given threshold), **log loss** and **ROC AUC** are good metrics*

to focus on. The numbers themselves are most useful in comparing different models, which is done in a later section.

Understanding the Logistic regression model fit

Now that you've fit the logistic regression model, you can use BigQuery ML's weights functionality to see the model coefficients for each predictor.

1. In the query **EDITOR**, click **COMPOSE NEW QUERY**.
2. Copy and paste the following code into the query **EDITOR**. This is a simple call to the **ML.WEIGHTS** function to the logistic regression model that you fit in the previous step.

```
SELECT
 *
FROM
 ML.WEIGHTS(MODEL soccer.xg_logistic_reg_model)
;
```

3. Click **Run**. The results are displayed below the query window.

Query results SAVE RESULTS EXPLORE DATA

Query complete (0.3 sec elapsed, 124 B processed)

Job information | Results | JSON | Execution details

Row	processed_input	weight	category_weights.category	category_weights.weight
1	shotType	*null*	Free kick shot	-0.9095405446137811
			Shot	-1.296812234577563
			Penalty	1.148237947578076
2	shotDistance	-0.05107529365432086		
3	shotAngle	0.03483486523751007		
4	__INTERCEPT__	-1.0885489917713715		

There are weights for each input - a single numerical weight for the continuous features, 1 value per potential category for the categorical features - that correspond to coefficients in a logistic regression model. It's sometimes dangerous to interpret

logistic regression model coefficients directly because of correlation among the predictors, among other things.

At minimum, the sign of these seem reasonable based on general subject matter knowledge of soccer:

- Penalty kicks have a much higher chance of success than other shot types
- Shots at further distance have less chance of success
- Shots with greater angles to the goal have higher chance of success

Seeing these results can provide some confidence that the model is doing something logical.

Create a boosted tree Model for expected goals

Next, you will fit a boosted tree model to the data using BigQuery ML's implementation of XGBoost, and compare the results to the previously fit logistic regression model. In theory, boosted trees can be more accurate because of their ability to take into account nonlinear relationships between features and the outcome as well as interactions among features.

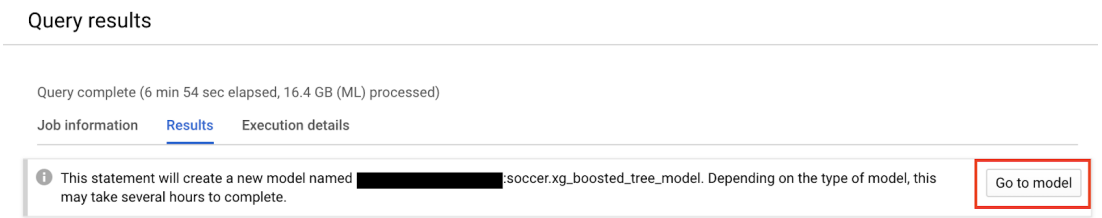
1. In the query **Editor**, click **Compose New Query**.
2. Copy and paste the following code into the query **Editor**.

```
CREATE MODEL `soccer.xg_boosted_tree_model`
OPTIONS(
 model_type = 'BOOSTED_TREE_CLASSIFIER',
 input_label_cols = ['isGoal']
 ) AS
SELECT
 Events.subEventName AS shotType,
 /* 101 is known Tag for 'goals' from goals table */
 (101 IN UNNEST(Events.tags.id)) AS isGoal,
  `soccer.GetShotDistanceToGoal`(Events.positions[ORDINAL(1)].x,
   Events.positions[ORDINAL(1)].y) AS shotDistance,
 `soccer.GetShotAngleToGoal`(Events.positions[ORDINAL(1)].x,
   Events.positions[ORDINAL(1)].y) AS shotAngle
FROM
 `soccer.events` Events
```

```
LEFT JOIN
 `soccer.matches` Matches ON
   Events.matchId = Matches.wyId
LEFT JOIN
 `soccer.competitions` Competitions ON
   Matches.competitionId = Competitions.wyId
WHERE
 /* Filter out World Cup matches for model fitting purposes */
 Competitions.name != 'World Cup' AND
 /* Includes both "open play" & free kick shots (including penalties) */
 (
   eventName = 'Shot' OR
   (eventName = 'Free Kick' AND subEventName IN ('Free kick shot', 'Penalty'))
 )
;
```

The SQL statement is identical to the one to fit the logistic regression above, except for changes in the model type to pick a boosted tree classifier and the name of the model object created.

3. Click **Run**.
4. The **Query results** section should display a message saying This statement will create a new model named: **<PROJECT_NAME>:soccer.xg_boosted_tree_model**. Depending on the type of model, this may take several hours to complete. In this case, it should take several minutes to finish.

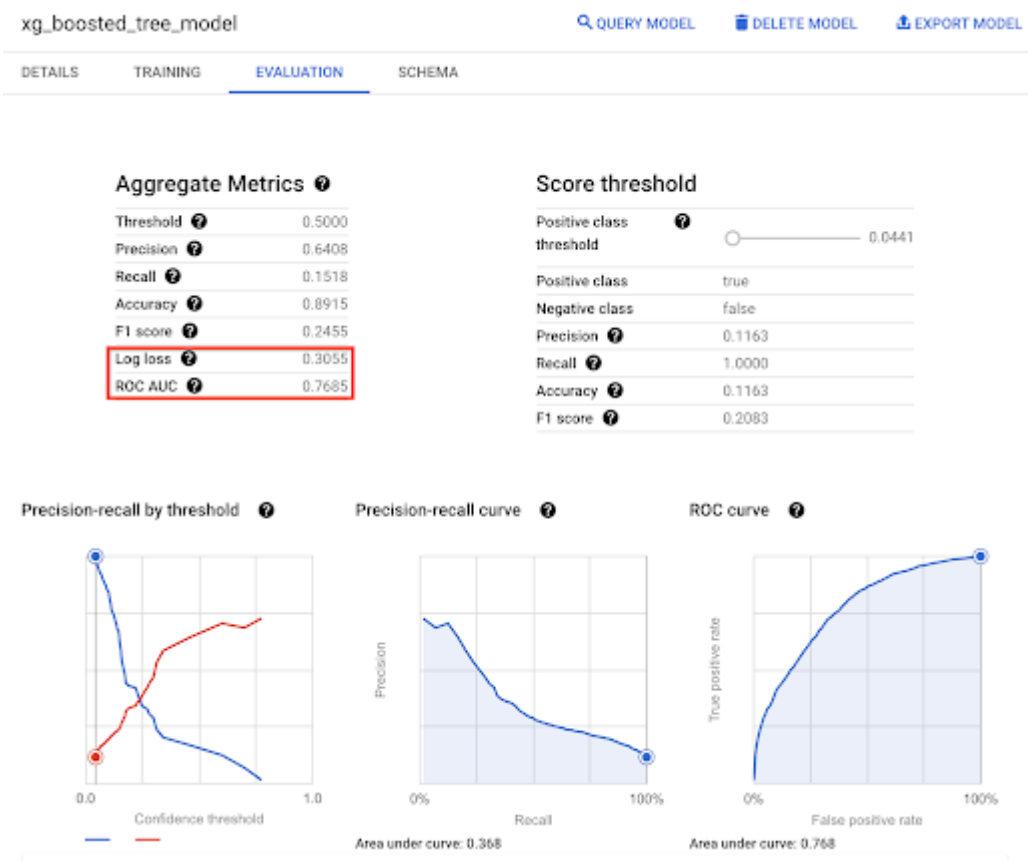


5. Once the model is done training, look for a "Query complete" notification in the **Query results** section.
6. Click on **Go to model** at the far right next to the message about model creation.

This will open up a new tab that has information about the model that was just trained.

Click to the **EVALUATION** tab and take a look at some of the metrics, particularly **"Log loss"** and **"ROC AUC"** under **Aggregate Metrics**.

***NOTE**: Results may vary slightly based on the inherent randomness in the model fitting procedure.*



The log loss and ROC AUC of the boosted tree model is pretty similar to that from the logistic regression model, but slightly worse (higher log loss and lower ROC AUC).

In this section BigQuery was used to fit machine learning models using soccer data. An expected goals model can be created in a variety of ways.In the next section learn how to apply an expected goals model to new data using BigQuery ML.

Apply an expected goals model to new data

Now that you've fit an expected goals model of reasonable accuracy and explainability, you can apply it to "new" data - in this case, the 2018 World Cup (which was left out of the model fitting).
The logistic regression model created in the previous step will be used to assess the difficulty of each shot and goal in that competition, enabling identification of the most "impressive" goals in the tournament.

Get probabilities for all shots in 2018 World Cup

You can use BigQuery ML's prediction functionality with the logistic regression model fit in the previous step (or even the boosted tree model, if you prefer) to look at the probability of each shot scoring or not in the World Cup.

1. In the query Editor, click Compose New Query.
2. Copy and paste the following code into the query Editor.

```
SELECT
 *
FROM
 ML.PREDICT(
   MODEL `soccer.xg_logistic_reg_model`,
   (
     SELECT
       Events.subEventName AS shotType,
       /* 101 is known Tag for 'goals' from goals table */
       (101 IN UNNEST(Events.tags.id)) AS isGoal,

       `soccer.GetShotDistanceToGoal`(Events.positions[ORDINAL(1)].x,
           Events.positions[ORDINAL(1)].y) AS shotDistance,
       `soccer.GetShotAngleToGoal`(Events.positions[ORDINAL(1)].x,
           Events.positions[ORDINAL(1)].y) AS shotAngle
     FROM
       `soccer.events` Events
     LEFT JOIN
       `soccer.matches` Matches ON
           Events.matchId = Matches.wyId
     LEFT JOIN
       `soccer.competitions` Competitions ON
```

```
            Matches.competitionId = Competitions.wyId
        WHERE
          /* Look only at World Cup matches for model predictions */
          Competitions.name = 'World Cup' AND
          /* Includes both "open play" & free kick shots (including penalties)
*/
          (
              eventName = 'Shot' OR
              (eventName = 'Free Kick' AND subEventName IN ('Free kick shot',
'Penalty'))
          )
      )
    )
```

This calls the ML.PREDICT function on the logistic regression model that was fit previously. The SELECT statement creates the same fields on all shots from the events data as the ones used to fit the model, but this time it filters to only World Cup matches (instead of leaving them out) since those are the shots the model is to be applied to.

3. Click Run. The results are displayed below the query window.

Query results SAVE RESULTS EXPLORE DATA

Query complete (1.3 sec elapsed, 228.1 MB processed)

Job information Results JSON Execution details

Row	predicted_isGoal	predicted_isGoal_probs.label	predicted_isGoal_probs.prob	shotType	isGoal	shotDistance	shotAngle
1	true	true	0.7019285800667375	Penalty	true	10.521996008362672	38.30138598528921
		false	0.29807141993326247				
2	false	true	0.06485857421405333	Free kick shot	false	24.280864893985964	16.355831431767868
		false	0.9351414257859467				
3	false	true	0.041021252337121165	Free kick shot	false	30.809225890956757	12.054048768557273
		false	0.9589787476628788				
4	true	true	0.6659905919099712	Penalty	false	11.57	35.061383677285804
		false	0.33400940809002877				
5	false	true	0.03908415691092193	Free kick shot	false	31.462669943919256	11.565567946050386
		false	0.9609158430890781				

The output shows a couple of prediction-related fields along with the original fields in that data of shot type, distance, angle, and success:

- predicted_isGoal, a binary prediction of whether the shot would result in a goal or not
- predicted_isGoal_probs, an array of (label, prob) pairs that shows the probability of each shot being successful and not from the model

Identify most unlikely goals using model probabilities

Building on the previous section that yielded probabilities for all shots in the 2018 World Cup, use BigQuery to extract the probability of each goal, then sort from lowest to highest to see the most unlikely goals scored in the World Cup based on the factors in the model.

Theoretically, this should yield a list of some of the most impressive goals in the tournament, at least based on the factors in the model (distance, angle, etc.).

1. In the query **Editor**, click **Compose New Query**.
2. Add the following code into the query **Editor**.

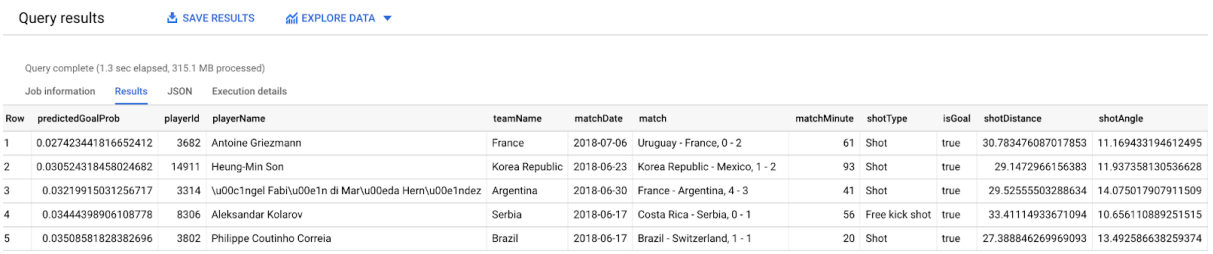
```
SELECT
 predicted_isGoal_probs[ORDINAL(1)].prob AS predictedGoalProb,
 * EXCEPT (predicted_isGoal, predicted_isGoal_probs),
FROM
 ML.PREDICT(
   MODEL `soccer.xg_logistic_reg_model`,
   (
     SELECT
       Events.playerId,
       (Players.firstName || ' ' || Players.lastName) AS playerName,
       Teams.name AS teamName,
       CAST(Matches.dateutc AS DATE) AS matchDate,
       Matches.label AS match,
     /* Convert match period and event seconds to minute of match */
       CAST((CASE
         WHEN Events.matchPeriod = '1H' THEN 0
         WHEN Events.matchPeriod = '2H' THEN 45
         WHEN Events.matchPeriod = 'E1' THEN 90
         WHEN Events.matchPeriod = 'E2' THEN 105
         ELSE 120
         END) +
         CEILING(Events.eventSec / 60) AS INT64)
         AS matchMinute,
       Events.subEventName AS shotType,
       /* 101 is known Tag for 'goals' from goals table */
       (101 IN UNNEST(Events.tags.id)) AS isGoal,

       `soccer.GetShotDistanceToGoal`(Events.positions[ORDINAL(1)].x,
           Events.positions[ORDINAL(1)].y) AS shotDistance,
       `soccer.GetShotAngleToGoal`(Events.positions[ORDINAL(1)].x,
```

```
        Events.positions[ORDINAL(1)].y) AS shotAngle
    FROM
      `soccer.events` Events
    LEFT JOIN
      `soccer.matches` Matches ON
        Events.matchId = Matches.wyId
    LEFT JOIN
      `soccer.competitions` Competitions ON
        Matches.competitionId = Competitions.wyId
    LEFT JOIN
      `soccer.players` Players ON
        Events.playerId = Players.wyId
    LEFT JOIN
      `soccer.teams` Teams ON
        Events.teamId = Teams.wyId
    WHERE
      /* Look only at World Cup matches to apply model */
      Competitions.name = 'World Cup' AND
      /* Includes both "open play" & free kick shots (but not penalties) */
      (
        eventName = 'Shot' OR
        (eventName = 'Free Kick' AND subEventName IN ('Free kick shot'))
      ) AND
      /* Filter only to goals scored */
      (101 IN UNNEST(Events.tags.id))
  )
)
ORDER BY
  predictedgoalProb
```

This builds on the ML.PREDICT function call in the previous section, filtering to only shots that were goals (excluding penalties, since the goal is to get the most "impressive" goals), extracting only the probability of a goal being scored from the predictions, and ordering by that.

A few fields are added in the SELECT statement from the events data and the other tables that can be joined to it to get other information of interest on each goal like the player and team who scored it, match date and information, and minute into the match of the goal.

3. Click Run. The results are displayed below the query window.



Query results SAVE RESULTS EXPLORE DATA

Query complete (1.3 sec elapsed, 315.1 MB processed)

Job information | Results | JSON | Execution details

Row	predictedGoalProb	playerId	playerName	teamName	matchDate	match	matchMinute	shotType	isGoal	shotDistance	shotAngle
1	0.027423441816652412	3682	Antoine Griezmann	France	2018-07-06	Uruguay - France, 0 - 2	61	Shot	true	30.783476087017853	11.169433194612495
2	0.030524318458024682	14911	Heung-Min Son	Korea Republic	2018-06-23	Korea Republic - Mexico, 1 - 2	93	Shot	true	29.1472966156383	11.937358130536628
3	0.03219915031256717	3314	\u00c1ngel Fabi\u00e1n di Mar\u00eda Hern\u00e1ndez	Argentina	2018-06-30	France - Argentina, 4 - 3	41	Shot	true	29.52555503288634	14.075017907911509
4	0.03444398906108778	8306	Aleksandar Kolarov	Serbia	2018-06-17	Costa Rica - Serbia, 0 - 1	56	Free kick shot	true	33.41114933671094	10.656110889251515
5	0.03508581828382696	3802	Philippe Coutinho Correia	Brazil	2018-06-17	Brazil - Switzerland, 1 - 1	20	Shot	true	27.388846269969093	13.492586638259374

The top few lines show the 2018 World Cup goals that the model believed had the least likely chance of going in - all with fairly long shot distances (near 30 meters) and relatively tight angles (between 10° and 15°).

For some visual validation, you can watch the Antoine Griezmann goal that earned the top spot on this list - to be fair, it's more of an unlikely goal than an impressive one based on how it was played by the goalkeeper.

For more of a "wow" goal in line with what this analysis intends to uncover, check out this amazing shot by Korea Republic's Son Heung-min vs Mexico that came in second in the results.

In the next section test your understanding of what you have learned in this introduction to BigQuery ML.