Machine Learning Engineer Nanodegree

Capstone Project Report

Ravi Kandikonda Jan 26th 2020

I. Definition

Project Background

Early childhood education research is conducted by many reputed organizations. PBS Kids is one such organization trying to gain deep insights into how modern media helps young children learn important skills needed for success in life. They use unique measuring games and measurement-focused video clips that feature well-known PBS characters to teach the concepts and also reward them as needed. Data Science Bowl (https://datasciencebowl.com/) is conducting a challenge on Kaggle in which participants have to predict certain scores that will aid PBS in devising higher quality educational media and improved learning processes. Of the many challenges presented on Kaggle, I chose this one due to my interest in early childhood education and providing the right kind of tools and resources to kids to help them relaize their potential sooner.

Problem Statement

The Kaggle challenge provides anonymous gameplay data which includes the information about the video clips that were watched by the kids and the games that they played while using the PBS Kids Measure Up! app. In this app the children who are aged between 3 and 5 learn early STEM concepts like length, width, capacity and weight while experiencing an adventure-like gameplay through Treetop City, Magma Peak and Crystal Caves. They collect rewards along the way and can unlock digital toys during their play. The challenge is to predict the scores on the in-game assessments and create an algorithm that will lead to better game design and improve learning outcomes. The challenge has been open for several weeks and they have published a live leaderboard online. The solution is scored using a quadratic weighted kappa, which measures the agreement between two outcomes.

More details about the scoring are available here. https://www.kaggle.com/c/data-science-bowl-2019/overview/evaluation

Dataset and Inputs

The data comes from the PBS Kids Measure Up! app. As the child uses the app to navigate a map while playing games, watching video clips and completing various levels, data is collected and anonymized in accordance with child privacy regulations. This is presented in a tabular form containing the child's interactions within the app, such as the in-app assessment score or their path through the game and certain analytics. The assessments are designed to test the child's comprehension of a certain set of

skills. The game presents five assessments: Bird Measurer, Cart Balancer, Cauldron Filler, Chest Sorter, and Mushroom Sorter.

The challenge provides two main data file (train.csv and test.csv) each of which contain the gameplay events with the following features:

event_id: Randomly generated unique identifier for the event type.

game_session: Randomly generated unique identifier grouping events within a single game or video play session.

timestamp: Client-generated datetime

event_data: Semi-structured JSON formatted string containing the events arameters. Default fields are: event_count, event_code, and game_time; therwise fields are determined by the event type.

installation_id: Randomly generated unique identifier grouping game sessions within a single installed application instance.

event_count: Incremental counter of events within a game session (offset at 1). Extracted from event_data.

event_code: Identifier of the event 'class'. Unique per game, but may be duplicated across games. E.g. event code '2000' always identifies the 'Start Game' event for all games. Extracted from event_data.

game_time: Time in milliseconds since the start of the game session. Extracted from event_data.

title: Title of the game or video.

type: Media type of the game or video. Possible values are: 'Game', 'Assessment', 'Activity', 'Clip'.

world: The section of the application the game or video belongs to. Helpful to identify the educational curriculum goals of the media. Possible values are: 'NONE' (at the app's start screen), TREETOPCITY' (Length/Height), 'MAGMAPEAK' (Capacity/Displacement), 'CRYSTALCAVES' (Weight).

The training set contains 11341042(~11.3M) records. It has 11 columns as shown in the figure below.

```
print(training_data.shape)
display(training_data.head(n=10))
```

```
(11341042, 11)
```

	event_id	game_session	timestamp	event_data	installation_id	event_count	event_code	game_time	title	type	world
0	27253bdc	45bb1e1b6b50c07b	2019-09-06T17:53:46.937Z	{"event_code": 2000, "event_count": 1}	0001e90f	1	2000	0	Welcome to Lost Lagoon!	Clip	NONE
1	27253bdc	17eeb7f223665f53	2019-09-06T17:54:17.519Z	{"event_code": 2000, "event_count": 1}	0001e90f	1	2000	0	Magma Peak - Level 1	Clip	MAGMAPEAK
2	77261ab5	0848ef14a8dc6892	2019-09-06T17:54:56.302Z	{"version":"1.0","event_count":1,"game_time":0...	0001e90f	1	2000	0	Sandcastle Builder (Activity)	Activity	MAGMAPEAK
3	b2dba42b	0848ef14a8dc6892	2019-09-06T17:54:56.387Z	{"description":"Let's build a sandcastle! Firs...	0001e90f	2	3010	53	Sandcastle Builder (Activity)	Activity	MAGMAPEAK
4	1bb5fbdb	0848ef14a8dc6892	2019-09-06T17:55:03.253Z	{"description":"Let's build a sandcastle! Firs...	0001e90f	3	3110	6972	Sandcastle Builder (Activity)	Activity	MAGMAPEAK
5	1325467d	0848ef14a8dc6892	2019-09-06T17:55:06.279Z	{"coordinates":{"x":583,"y":605,"stage_width":...	0001e90f	4	4070	9991	Sandcastle Builder (Activity)	Activity	MAGMAPEAK
6	1325467d	0848ef14a8dc6892	2019-09-06T17:55:06.913Z	{"coordinates":{"x":601,"y":570,"stage_width":...	0001e90f	5	4070	10622	Sandcastle Builder (Activity)	Activity	MAGMAPEAK
7	1325467d	0848ef14a8dc6892	2019-09-06T17:55:07.546Z	{"coordinates":{"x":250,"y":665,"stage_width":...	0001e90f	6	4070	11255	Sandcastle Builder (Activity)	Activity	MAGMAPEAK
8	1325467d	0848ef14a8dc6892	2019-09-06T17:55:07.979Z	{"coordinates":{"x":279,"y":629,"stage_width":...	0001e90f	7	4070	11689	Sandcastle Builder (Activity)	Activity	MAGMAPEAK
9	1325467d	0848ef14a8dc6892	2019-09-06T17:55:08.566Z	{"coordinates":{"x":839,"y":654,"stage_width":...	0001e90f	8	4070	12272	Sandcastle Builder (Activity)	Activity	MAGMAPEAK

The testing set contains 1156414(1.1M) records and has 11 columns as can be seen from the image below.

```
print(testing_data.shape)
display(testing_data.head(n=10))
```

```
(1156414, 11)
```

	event_id	game_session	timestamp	event_data	installation_id	event_count	event_code	game_time	title	type	world
0	27253bdc	0ea9ecc81a565215	2019-09-10T16:50:24.910Z	{"event_code": 2000, "event_count": 1}	00abaee7	1	2000	0	Welcome to Lost Lagoon!	Clip	NONE
1	27253bdc	c1ea43d8b8261d27	2019-09-10T16:50:55.503Z	{"event_code": 2000, "event_count": 1}	00abaee7	1	2000	0	Magma Peak - Level 1	Clip	MAGMAPEAK
2	27253bdc	7ed86c6b72e725e2	2019-09-10T16:51:51.805Z	{"event_code": 2000, "event_count": 1}	00abaee7	1	2000	0	Magma Peak - Level 2	Clip	MAGMAPEAK
3	27253bdc	7e516ace50e7fe67	2019-09-10T16:53:12.825Z	{"event_code": 2000, "event_count": 1}	00abaee7	1	2000	0	Crystal Caves - Level 1	Clip	CRYSTALCAVES
4	7d093bf9	a022c3f60ba547e7	2019-09-10T16:54:12.115Z	{"version":"1.0","round":0,"event_count":1,"ga...	00abaee7	1	2000	0	Chow Time	Game	CRYSTALCAVES
5	f93fc684	a022c3f60ba547e7	2019-09-10T16:54:14.338Z	{"coordinates":{"x":515,"y":697,"stage_width":...	00abaee7	2	4010	2232	Chow Time	Game	CRYSTALCAVES
6	7ec0c298	a022c3f60ba547e7	2019-09-10T16:54:16.553Z	{"description":"It's Chow Time! We have some V...	00abaee7	3	3010	4445	Chow Time	Game	CRYSTALCAVES
7	0d1da71f	a022c3f60ba547e7	2019-09-10T16:54:23.364Z	{"description":"It's Chow Time! We have some V...	00abaee7	4	3110	11251	Chow Time	Game	CRYSTALCAVES
8	63f13dd7	a022c3f60ba547e7	2019-09-10T16:54:23.365Z	{"dinosaur":"stacey","diet":"herbivore","targe...	00abaee7	5	2020	11251	Chow Time	Game	CRYSTALCAVES
9	7372e1a5	a022c3f60ba547e7	2019-09-10T16:54:25.029Z	{"coordinates":{"x":901,"y":449,"stage_width":...	00abaee7	6	4070	12929	Chow Time	Game	CRYSTALCAVES

The testing set does not specify the target variable (correct answers). To reduce overfitting, I will use KFold cross-validation and use quadratic weighted kappa to evaluate the model's fit using the generated score. The dataset contains another file specs.csv which gives the specification of the various event types. The dataset also includes a file train_labels.csv which demonstrates how to compute the ground truth for the assessments in the training set.

II. Data Analysis and Exploration

The entire project was developed and run on the kaggle platform and is available here.
https://www.kaggle.com/gr8ergood/2019-data-science-bowl-a-possible-solution

The inputs to the problem are a set of files provided by kaggle.com on its competition website

.

In[2]:

```
!ls -alh ../input/data-science-bowl-2019/
```

```
total 4.0G
drwxr-xr-x 2 nobody nogroup 4.0K Nov 26 23:56 .
drwxr-xr-x 3 root   root    4.0K Jan 26 21:51 ..
-rw-r--r-- 1 nobody nogroup  11K Nov 26 23:56 sample_submission.csv
-rw-r--r-- 1 nobody nogroup 400K Nov 26 23:56 specs.csv
-rw-r--r-- 1 nobody nogroup 380M Nov 26 23:57 test.csv
-rw-r--r-- 1 nobody nogroup 3.7G Nov 26 23:58 train.csv
-rw-r--r-- 1 nobody nogroup 1.1M Nov 26 23:56 train_labels.csv
```

These files contain data that has the following structure.

```
Train data has 11341042 rows and the following 11 columns.
['event_id', 'game_session', 'timestamp', 'event_data', 'installation_id', 'event_count', 'event_code', 'game_time', 'title', 'type', 'world']

Test data has 1156414 rows and the following 11 columns.
['event_id', 'game_session', 'timestamp', 'event_data', 'installation_id', 'event_count', 'event_code', 'game_time', 'title', 'type', 'world']

The specs data has 386 rows and the following 3 columns.
['event_id', 'info', 'args']

The labels data has 17690 rows and the following 7 columns.
['game_session', 'installation_id', 'title', 'num_correct', 'num_incorrect', 'accuracy', 'accuracy_group']

The sample submission has 1000 rows and the following 2 columns.
['installation_id', 'accuracy_group']
```

```
7734558 ids out of 11341042 in the training set are also in the labels.
There are a total of 3614 unique ids in the reduced_training_data.
There are a total of 3614 unique ids in the train_labels.
The number of common ids in the reduced set vs the labels is 3614
```

Taking a deeper look, I notice that there are 3614 unique installation ids in the training data which have their correct target variable in the train_labels dataset. We need to train a model on the training data and then validate against the ground truth provided in the labels.

```
print(train_labels.shape)
train_labels.head()
print("There are a total of {} unique game_sessions in the train_labels out of total of {}" \
      .format(len(train_labels['game_session'].unique()), train_labels.shape[0]))
```

```
(17690, 7)
There are a total of 17690 unique game_sessions in the train_labels out of total of 17690
```

From the above we can see that the target variable 'accuracy_group’ is provided for each unique combination of an installation_id+game_session. It helps to know this when we are compiling data from the training_data.

Now lets compile some data from the training dataset.

Data Preprocessing

In the pre-process step, I step through each game_session grouped by the installation id and collect the 'true' and 'false' attempts made by the child in each Assessment exercise. I tally them and calculate a total correct vs false and accumulated list of attempts and scores.

Full code can be accessed at the github link given below: https://github.com/gr8ergud/Udacity-MLND-Capstone-Project/blob/master/capstone.ipynb

The 'build_features' method translates the raw data into trainable data by extracting relevant features.

```
print("Length of train is {}".format(training_data.shape))
print("Length of test is {}".format(testing_data.shape))
reduce_train, reduce_test, categoricals = get_train_and_test(training_data, testing_data)
```

```
Length of train is (11341042, 12)
Length of test is (1156414, 12)
Train = 4242, test = 1000
```


```
Length of reduced_train is (17690, 891)
Length of reduced_test is (2347, 891)
```

I selected a notebook on teh kaggle.com website as a starting point and started making changes that I thought would be needed to get a benchmark, a solution and other analyses peformed. Some of the algorithms used in the source kernel were kept intact but I heavily modified it suit the nanodegree requirements. https://www.kaggle.com/braquino/convert-to-regression

Feature Engineering

To reduce or optimize teh number of features, I have searched for columns that have only '0' in the cells and removed those columns as they don't provide any value to the training.

```
# Extract features from the training data.
all_labels = reduce_train.columns
print("There a total of {} columns in the reduced training dataset".format(len(all_labels)))

# Remove the columns that appear in the sample_submission csv.
features = [x for x in all_labels if x not in sample_submission.columns]
print("There a total of {} columns in the feature set".format(len(features)))

# Remove columns with only '0' in the values
features = reduce_train.loc[:,(reduce_train.sum(axis=0) != 0)].columns # delete useless columns
print("{} columns were removed that contained only '0' in the cell. {} remaining" \
      .format(len(reduce_train.columns) - len(features), len(features)))
```

```
There a total of 891 columns in the reduced training dataset
There a total of 889 columns in the feature set
12 columns were removed that contained only '0' in the cell. 879 remaining
```

The above action resulted in around 850 labels in the trainbale dataset. All the categorical lables were already mapped and converted to numericals and also one-hot encoded during the feature extraction step.

Cross Validation

For cross validation I picked the StratifiedKFold cross validation and used n_folds=5 to reduce overfitting of the model.

Evaluation Metric

To compare the solution's model with a benchmark, the quadratic weighted kappa score was selected.

The outcomes in this competition are grouped into 4 groups (labeled `accuracy_group` in the data):

- 3: the assessment was solved on the first attempt
- 2: the assessment was solved on the second attempt
- 1: the assessment was solved after 3 or more attempts
- 0: the assessment was never solved

The quadratic weighted kappa is calculated as follows. First, an N x N histogram matrix *O* is constructed, such that *Oi,j* corresponds to the number of `installation_id` s *i* (actual) that received a predicted value *j*. An *N-by-N* matrix of weights, *w*, is calculated based on the difference between actual and predicted values:

$$w_{i,j} = \frac{(i-j)^2}{(N-1)^2}$$

An *N-by-N* histogram matrix of expected outcomes, *E*, is calculated assuming that there is no correlation between values. This is calculated as the outer product between the actual histogram vector of outcomes and the predicted histogram vector, normalized such that *E* and *O* have the same sum.

From these three matrices, the quadratic weighted kappa is calculated as:

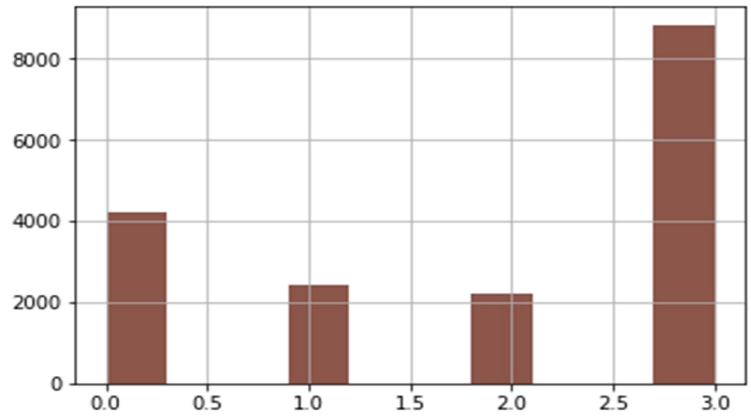
$$\kappa = 1 - \frac{\sum_{i,j} w_{i,j} O_{i,j}}{\sum_{i,j} w_{i,j} E_{i,j}}.$$

III. Benchmark

I picked the Logistic Regression model to benchmark against my solution. I trained the model on the same set of data that I used for the main solution. The full code is available here: https://github.com/gr8ergud/Udacity-MLND-Capstone-Project/blob/master/capstone.ipynb

The quadratic weighted kappa score for the benchmark turned out to be 0.237 (1 is full agreement with the ground truth). The distribution of the predicted scores for the test data is plotted in the graph below.

```
Our oof cohen kappa score is:  0.23610529554826576
```



IV. Solution

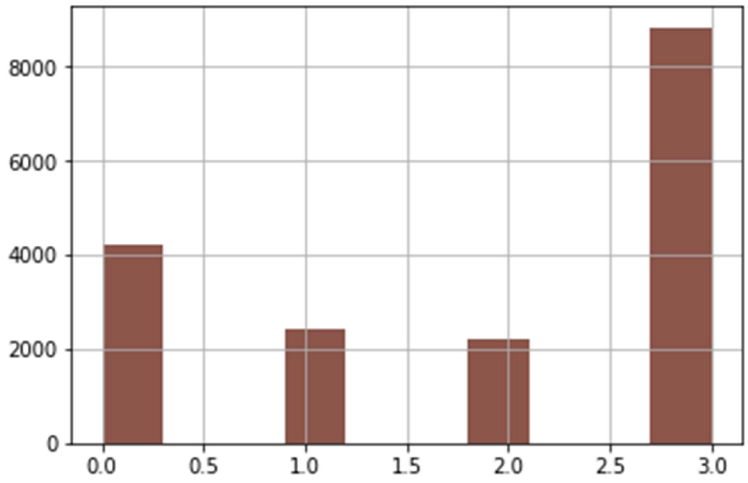
The entire project was developed and run on the kaggle platform and is available here. https://www.kaggle.com/gr8ergood/2019-data-science-bowl-a-possible-solution For the solution I selected the LightGBM estimator which has a few advantages over the XGBoost and other neural network based architectures in speed, memory usage etc. The LightGBM model was parameterized as shown below and trained on the extracted dataset with expanded labels.

```
class Lgb_Model(Base_Model):

    def train_model(self, train_set, val_set):
        verbosity = 100 if self.verbose else 0
        return lgb.train(self.params, train_set, valid_sets=[train_set, val_set], verbose_eval=verbosity)

    def convert_dataset(self, x_train, y_train, x_val, y_val):
        train_set = lgb.Dataset(x_train, y_train, categorical_feature=self.categoricals)
        val_set = lgb.Dataset(x_val, y_val, categorical_feature=self.categoricals)
        return train_set, val_set

    def get_params(self):
        params = {'n_estimators':5000,
                  'boosting_type': 'gbdt',
                  'objective': 'regression',
                  'metric': 'rmse',
                  'subsample': 0.75,
                  'subsample_freq': 1,
                  'learning_rate': 0.01,
                  'feature_fraction': 0.9,
                  'max_depth': 15,
                  'lambda_l1': 1,
                  'lambda_l2': 1,
                  'early_stopping_rounds': 100
                  }
        return params
```

V. Results

The training of the data using the LightGBM model resulted in a higher quadratic weighted kappa score compared to the benchmark.

```
Our oof cohen kappa score is:  0.592203757512775
```



The model achieved a score of **0.592** which is greater than the benchmark score of **0.237**.

The original kernel used a combination of LGBM, CNN, XGBoost and later combined the predictions from each model with a certain weight. Obviously my solution is much simpler in order to satisfy the course requirement and there is a lot of room for improvement. Especially the hyperparameter tuning can be applied to improve the model.

VI. Conclusion

The selected problem of predicting number of attempts a child takes to answer an assessment question correctly is very interesting challenge. If we are able to predict which lessons have the most learning rate, educators can focus on improving those and also provide them a wider range of kids in the country. The LightGBM model is one of the better models but an XGBoost or a CNN model can also be used if the hyperparameters are properly tuned. My solution achieved a score of 0.592 which ranks highly on the leaderboard of the competition on kaggle.com website.