Quick and dirty regression

Python notebook using data from multiple data sources · 19,549 views · 1mo ago · beginner, feature engineering, regression, +1 more

397 Copy and Edit 1257

Version 3

Submission	Public Score
Ran successfully Submitted by Andrew Lukyanenko a month ago	0.536

Notebook Data Output Comments

General information

In this kernel I introduce a regression approach to this task. There were already several competitions on kaggle with kappa metric. Usually a regression approach with thresholds worked the best.

I use the code for feature generation from this kernel: https://www.kaggle.com/braquino/890-features (https://www.kaggle.com/braquino/890-features) And modelling is taken from my previous kernel. The code was changed to regression quite fast, so it may look not nice for now.

Importing libraries

Code

```
Using TensorFlow backend.
```

Helper functions and classes

Code

Code

Code

Code

Data overview

Let's have a look at the data at first.

In [6]:

```
def read_data():
    print('Reading train.csv file....')
    train = pd.read_csv('/kaggle/input/data-science-bowl-2019/train.csv')
    print('Training.csv file have {} rows and {} columns'.format(train.shape[0], train.shape[1]))

    print('Reading test.csv file....')
    test = pd.read_csv('/kaggle/input/data-science-bowl-2019/test.csv')
    print('Test.csv file have {} rows and {} columns'.format(test.shape[0], test.shape[1]))

    print('Reading train_labels.csv file....')
    train_labels = pd.read_csv('/kaggle/input/data-science-bowl-2019/train_labels.csv')
    print('Train_labels.csv file have {} rows and {} columns'.format(train_labels.shape[0], train_labels.shape[1]))

    print('Reading specs.csv file....')
    specs = pd.read_csv('/kaggle/input/data-science-bowl-2019/specs.csv')
    print('Specs.csv file have {} rows and {} columns'.format(specs
```

```
    print('Specs.csv file have {} rows and {} columns'.format(specs.
shape[0], specs.shape[1]))

    print('Reading sample_submission.csv file....')
    sample_submission = pd.read_csv('/kaggle/input/data-science-bowl
-2019/sample_submission.csv')
    print('Sample_submission.csv file have {} rows and {} columns'.f
ormat(sample_submission.shape[0], sample_submission.shape[1]))
    return train, test, train_labels, specs, sample_submission

def encode_title(train, test, train_labels):
    # encode title
    train['title_event_code'] = list(map(lambda x, y: str(x) + '_' +
str(y), train['title'], train['event_code']))
    test['title_event_code'] = list(map(lambda x, y: str(x) + '_' +
str(y), test['title'], test['event_code']))
    all_title_event_code = list(set(train["title_event_code"].unique
()).union(test["title_event_code"].unique()))
    # make a list with all the unique 'titles' from the train and tes
t set
    list_of_user_activities = list(set(train['title'].unique()).unio
n(set(test['title'].unique())))
    # make a list with all the unique 'event_code' from the train and
test set
    list_of_event_code = list(set(train['event_code'].unique()).unio
n(set(test['event_code'].unique())))
    list_of_event_id = list(set(train['event_id'].unique()).union(se
t(test['event_id'].unique())))
    # make a list with all the unique worlds from the train and test
 set
    list_of_worlds = list(set(train['world'].unique()).union(set(tes
t['world'].unique())))
    # create a dictionary numerating the titles
    activities_map = dict(zip(list_of_user_activities, np.arange(len
(list_of_user_activities))))
    activities_labels = dict(zip(np.arange(len(list_of_user_activiti
es)), list_of_user_activities))
    activities_world = dict(zip(list_of_worlds, np.arange(len(list_o
f_worlds))))
    assess_titles = list(set(train[train['type'] == 'Assessment']['t
itle'].value_counts().index).union(set(test[test['type'] == 'Assessm
ent']['title'].value_counts().index)))
    # replace the text titles with the number titles from the dict
    train['title'] = train['title'].map(activities_map)
    test['title'] = test['title'].map(activities_map)
    train['world'] = train['world'].map(activities_world)
    test['world'] = test['world'].map(activities_world)
    train_labels['title'] = train_labels['title'].map(activities_map
)
    win_code = dict(zip(activities_map.values(), (4100*np.ones(len(a
ctivities_map))).astype('int')))
    # then, it set one element, the 'Bird Measurer (Assessment)' as 4
110, 10 more than the rest
    win_code[activities_map['Bird Measurer (Assessment)']] = 4110
    # convert text into datetime
    train['timestamp'] = pd.to_datetime(train['timestamp'])
    test['timestamp'] = pd.to_datetime(test['timestamp'])


    return train, test, train_labels, win_code, list_of_user_activit
ies, list_of_event_code, activities_labels, assess_titles, list_of_e
```

```
ies, list_of_event_code, activities_labels, assess_titles, list_of_e
vent_id, all_title_event_code

def get_data(user_sample, test_set=False):
    '''
    The user_sample is a DataFrame from train or test where the only
 one
    installation_id is filtered
    And the test_set parameter is related with the labels processing,
that is only requered
    if test_set=False
    '''
    # Constants and parameters declaration
    last_activity = 0

    user_activities_count = {'Clip':0, 'Activity': 0, 'Assessment':
0, 'Game':0}

    # new features: time spent in each activity
    last_session_time_sec = 0
    accuracy_groups = {0:0, 1:0, 2:0, 3:0}
    all_assessments = []
    accumulated_accuracy_group = 0
    accumulated_accuracy = 0
    accumulated_correct_attempts = 0
    accumulated_uncorrect_attempts = 0
    accumulated_actions = 0
    counter = 0
    time_first_activity = float(user_sample['timestamp'].values[0])
    durations = []
    last_accuracy_title = {'acc_' + title: -1 for title in assess_ti
tles}
    event_code_count: Dict[str, int] = {ev: 0 for ev in list_of_even
t_code}
    event_id_count: Dict[str, int] = {eve: 0 for eve in list_of_even
t_id}
    title_count: Dict[str, int] = {eve: 0 for eve in activities_labe
ls.values()}
    title_event_code_count: Dict[str, int] = {t_eve: 0 for t_eve in
all_title_event_code}

    # itarates through each session of one instalation_id
    for i, session in user_sample.groupby('game_session', sort=False
):
        # i = game_session_id
        # session is a DataFrame that contain only one game_session

        # get some sessions information
        session_type = session['type'].iloc[0]
        session_title = session['title'].iloc[0]
        session_title_text = activities_labels[session_title]



        # for each assessment, and only this kind off session, the fe
atures below are processed
        # and a register are generated
        if (session_type == 'Assessment') & (test_set or len(session
)>1):
            # search for event_code 4100, that represents the assessm
ents trial
            all_attempts = session.query(f'event_code == {win_code[s
```

```
ession_title]}')
            # then, check the numbers of wins and the number of losse
s
            true_attempts = all_attempts['event_data'].str.contains(
'true').sum()
            false_attempts = all_attempts['event_data'].str.contains
('false').sum()
            # copy a dict to use as feature template, it's initialize
d with some itens:
            # {'Clip':0, 'Activity': 0, 'Assessment': 0, 'Game':0}
            features = user_activities_count.copy()
            features.update(last_accuracy_title.copy())
            features.update(event_code_count.copy())
            features.update(event_id_count.copy())
            features.update(title_count.copy())
            features.update(title_event_code_count.copy())
            features.update(last_accuracy_title.copy())

            # get installation_id for aggregated features
            features['installation_id'] = session['installation_id']
.iloc[-1]
            # add title as feature, remembering that title represents
the name of the game
            features['session_title'] = session['title'].iloc[0]
            # the 4 lines below add the feature of the history of the
trials of this player
            # this is based on the all time attempts so far, at the m
oment of this assessment
            features['accumulated_correct_attempts'] = accumulated_c
orrect_attempts
            features['accumulated_uncorrect_attempts'] = accumulated
_uncorrect_attempts
            accumulated_correct_attempts += true_attempts
            accumulated_uncorrect_attempts += false_attempts
            # the time spent in the app so far
            if durations == []:
                features['duration_mean'] = 0
            else:
                features['duration_mean'] = np.mean(durations)
            durations.append((session.iloc[-1, 2] - session.iloc[0,
2] ).seconds)
            # the accurace is the all time wins divided by the all ti
me attempts
            features['accumulated_accuracy'] = accumulated_accuracy/
counter if counter > 0 else 0
            accuracy = true_attempts/(true_attempts+false_attempts)
if (true_attempts+false_attempts) != 0 else 0
            accumulated_accuracy += accuracy
            last_accuracy_title['acc_' + session_title_text] = accur
acy
            # a feature of the current accuracy categorized
            # it is a counter of how many times this player was in ea
ch accuracy group
            if accuracy == 0:
                features['accuracy_group'] = 0
            elif accuracy == 1:
                features['accuracy_group'] = 3
            elif accuracy == 0.5:
                features['accuracy_group'] = 2
            else:
```

```
                features['accuracy_group'] = 1
            features.update(accuracy_groups)
            accuracy_groups[features['accuracy_group']] += 1
            # mean of the all accuracy groups of this player
            features['accumulated_accuracy_group'] = accumulated_accuracy_group/counter if counter > 0 else 0
            accumulated_accuracy_group += features['accuracy_group']
            # how many actions the player has done so far, it is initialized as 0 and updated some lines below
            features['accumulated_actions'] = accumulated_actions

            # there are some conditions to allow this features to be inserted in the datasets
            # if it's a test set, all sessions belong to the final dataset
            # it it's a train, needs to be passed throught this clausule: session.query(f'event_code == {win_code[session_title]}')
            # that means, must exist an event_code 4100 or 4110
            if test_set:
                all_assessments.append(features)
            elif true_attempts+false_attempts > 0:
                all_assessments.append(features)

            counter += 1

        # this piece counts how many actions was made in each event_code so far
        def update_counters(counter: dict, col: str):
                num_of_session_count = Counter(session[col])
                for k in num_of_session_count.keys():
                    x = k
                    if col == 'title':
                        x = activities_labels[k]
                    counter[x] += num_of_session_count[k]
                return counter

        event_code_count = update_counters(event_code_count, "event_code")
        event_id_count = update_counters(event_id_count, "event_id")
        title_count = update_counters(title_count, 'title')
        title_event_code_count = update_counters(title_event_code_count, 'title_event_code')

        # counts how many actions the player has done so far, used in the feature of the same name
        accumulated_actions += len(session)
        if last_activity != session_type:
            user_activities_count[session_type] += 1
            last_activitiy = session_type

    # if it't the test_set, only the last assessment must be predicted, the previous are scraped
    if test_set:
        return all_assessments[-1]
    # in the train_set, all assessments goes to the dataset
    return all_assessments

def get_train_and_test(train, test):
    compiled_train = []
    compiled_test = []
```

```
    for i, (ins_id, user_sample) in tqdm(enumerate(train.groupby('in
stallation_id', sort = False)), total = 17000):
        compiled_train += get_data(user_sample)
    for ins_id, user_sample in tqdm(test.groupby('installation_id',
sort = False), total = 1000):
        test_data = get_data(user_sample, test_set = True)
        compiled_test.append(test_data)
    reduce_train = pd.DataFrame(compiled_train)
    reduce_test = pd.DataFrame(compiled_test)
    categoricals = ['session_title']
    return reduce_train, reduce_test, categoricals

# read data
train, test, train_labels, specs, sample_submission = read_data()
# get usefull dict with maping encode
train, test, train_labels, win_code, list_of_user_activities, list_o
f_event_code, activities_labels, assess_titles, list_of_event_id, al
l_title_event_code = encode_title(train, test, train_labels)
# tranform function to get the train and test set
reduce_train, reduce_test, categoricals = get_train_and_test(train,
test)
```

```
Reading train.csv file....
Training.csv file have 11341042 rows and 11 columns
Reading test.csv file....
Test.csv file have 1156414 rows and 11 columns
Reading train_labels.csv file....
Train_labels.csv file have 17690 rows and 7 columns
Reading specs.csv file....
Specs.csv file have 386 rows and 3 columns
Reading sample_submission.csv file....
Sample_submission.csv file have 1000 rows and 2 columns


100%|██████████| 17000/17000 [07:40<00:00, 36.96it/s]
100%|██████████| 1000/1000 [00:50<00:00, 19.64it/s]
```

In [7]:

```
def preprocess(reduce_train, reduce_test):
    for df in [reduce_train, reduce_test]:
        df['installation_session_count'] = df.groupby(['installation
_id'])['Clip'].transform('count')
        df['installation_duration_mean'] = df.groupby(['installation
_id'])['duration_mean'].transform('mean')
        #df['installation_duration_std'] = df.groupby(['installation_
id'])['duration_mean'].transform('std')
        df['installation_title_nunique'] = df.groupby(['installation
_id'])['session_title'].transform('nunique')

        df['sum_event_code_count'] = df[[2050, 4100, 4230, 5000, 423
5, 2060, 4110, 5010, 2070, 2075, 2080, 2081, 2083, 3110, 4010, 3120,
3121, 4020, 4021,
                                        4022, 4025, 4030, 4031, 3010
, 4035, 4040, 3020, 3021, 4045, 2000, 4050, 2010, 2020, 4070, 2025,
2030, 4080, 2035,
                                        2040, 4090, 4220, 4095]].sum
(axis = 1)

        df['installation_event_code_count_mean'] = df.groupby(['inst
```

```
allation_id'])['sum_event_code_count'].transform('mean')
        #df['installation_event_code_count_std'] = df.groupby(['insta
llation_id'])['sum_event_code_count'].transform('std')

    features = reduce_train.loc[(reduce_train.sum(axis=1) != 0), (re
duce_train.sum(axis=0) != 0)].columns # delete useless columns
    features = [x for x in features if x not in ['accuracy_group',
'installation_id']] + ['acc_' + title for title in assess_titles]

    return reduce_train, reduce_test, features
# call feature engineering function
reduce_train, reduce_test, features = preprocess(reduce_train, reduc
e_test)
```

In [8]:

```
params = {'n_estimators':2000,
            'boosting_type': 'gbdt',
            'objective': 'regression',
            'metric': 'rmse',
            'subsample': 0.75,
            'subsample_freq': 1,
            'learning_rate': 0.04,
            'feature_fraction': 0.9,
         'max_depth': 15,
            'lambda_l1': 1,
            'lambda_l2': 1,
            'verbose': 100,
            'early_stopping_rounds': 100, 'eval_metric': 'cappa'
            }
```

In [9]:

```
y = reduce_train['accuracy_group']
```

In [10]:

```
n_fold = 5
folds = GroupKFold(n_splits=n_fold)
```

In [11]:

```
cols_to_drop = ['game_session', 'installation_id', 'timestamp', 'acc
uracy_group', 'timestampDate']
```

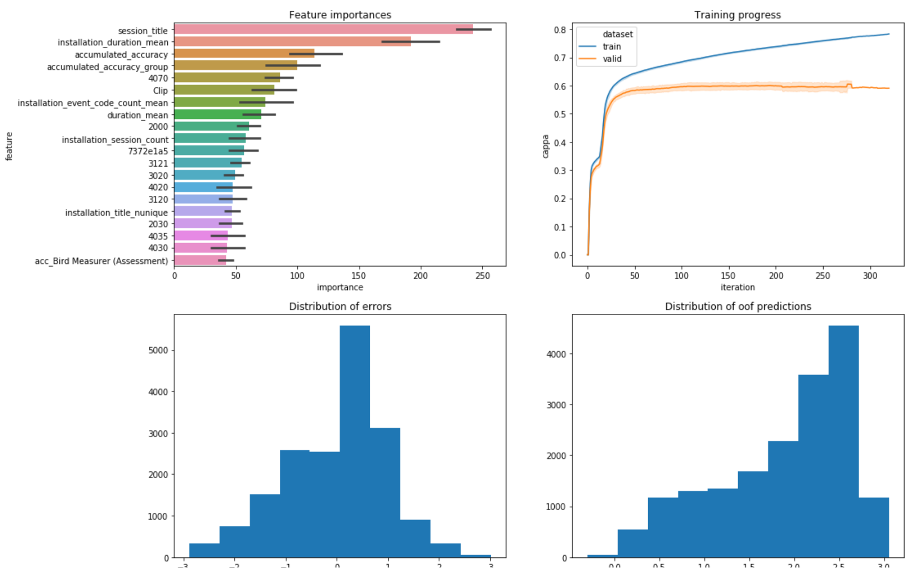
In [12]:

```
mt = MainTransformer()
ft = FeatureTransformer()
transformers = {'ft': ft}
regressor_model1 = RegressorModel(model_wrapper=LGBWrapper_regr())
regressor_model1.fit(X=reduce_train, y=y, folds=folds, params=params
, preprocesser=mt, transformers=transformers,
                    eval_metric='cappa', cols_to_drop=cols_to_drop)
```

```
Fold 1 started at Thu Nov 21 15:22:59 2019
Training until validation scores don't improve for 100 rounds
[100]   train's rmse: 0.907379  train's cappa: 0.680248 valid's rms
e: 0.977318     valid's cappa: 0.615109
[200]   train's rmse: 0.840297  train's cappa: 0.735068 valid's rms
e: 0.969994     valid's cappa: 0.619804
Early stopping, best iteration is:
```

```
[181]   train's rmse: 0.851173  train's cappa: 0.728361 valid's rms
e: 0.970069     valid's cappa: 0.622275
Fold 2 started at Thu Nov 21 15:23:25 2019
Training until validation scores don't improve for 100 rounds
[100]   train's rmse: 0.907134  train's cappa: 0.679167 valid's rms
e: 0.988913     valid's cappa: 0.612211
[200]   train's rmse: 0.841116  train's cappa: 0.734855 valid's rms
e: 0.980797     valid's cappa: 0.614212
Early stopping, best iteration is:
[107]   train's rmse: 0.901263  train's cappa: 0.684133 valid's rms
e: 0.987267     valid's cappa: 0.616905
Fold 3 started at Thu Nov 21 15:23:43 2019
Training until validation scores don't improve for 100 rounds
[100]   train's rmse: 0.907212  train's cappa: 0.68477  valid's rms
e: 0.976871     valid's cappa: 0.589914
[200]   train's rmse: 0.840906  train's cappa: 0.736571 valid's rms
e: 0.972553     valid's cappa: 0.591963
Early stopping, best iteration is:
[168]   train's rmse: 0.858943  train's cappa: 0.723588 valid's rms
e: 0.971449     valid's cappa: 0.592035
Fold 4 started at Thu Nov 21 15:24:05 2019
Training until validation scores don't improve for 100 rounds
[100]   train's rmse: 0.899214  train's cappa: 0.686836 valid's rms
e: 1.00162      valid's cappa: 0.588022
[200]   train's rmse: 0.833621  train's cappa: 0.738549 valid's rms
e: 0.99759      valid's cappa: 0.594229
[300]   train's rmse: 0.786116  train's cappa: 0.775862 valid's rms
e: 0.99772      valid's cappa: 0.592345
Early stopping, best iteration is:
[221]   train's rmse: 0.822675  train's cappa: 0.748087 valid's rms
e: 0.996906     valid's cappa: 0.596356
Fold 5 started at Thu Nov 21 15:24:31 2019
Training until validation scores don't improve for 100 rounds
[100]   train's rmse: 0.897422  train's cappa: 0.686783 valid's rms
e: 1.01423      valid's cappa: 0.57446
[200]   train's rmse: 0.830992  train's cappa: 0.742513 valid's rms
e: 1.01127      valid's cappa: 0.575817
Early stopping, best iteration is:
[176]   train's rmse: 0.844332  train's cappa: 0.730742 valid's rms
e: 1.01128      valid's cappa: 0.579765

CV mean score on train: 0.7230 +/- 0.0211 std.
CV mean score on valid: 0.6015 +/- 0.0159 std.
```



Making predictions

The preprocessing is a class, which was initially written by Abhishek Thakur here: https://www.kaggle.com/c/petfinder-adoption-prediction/discussion/76107 (https://www.kaggle.com/c/petfinder-adoption-prediction/discussion/76107) and later improved here https://www.kaggle.com/naveenasaithambi/optimizedrounder-improved (https://www.kaggle.com/naveenasaithambi/optimizedrounder-improved) (the improvement is is speed).

It can be used to find optimal coefficients for thresholds. In this kernel I'll show an example, but when you do it, don't forget a proper validation.

In [13]:

```
from functools import partial
import scipy as sp
class OptimizedRounder(object):
    """
    An optimizer for rounding thresholds
    to maximize Quadratic Weighted Kappa (QWK) score
    # https://www.kaggle.com/naveenasaithambi/optimizedrounder-improv
ed
    """
    def __init__(self):
        self.coef_ = 0

    def _kappa_loss(self, coef, X, y):
        """
        Get loss according to
        using current coefficients

        :param coef: A list of coefficients that will be used for rou
nding
        :param X: The raw predictions
        :param y: The ground truth labels
        """
        X_p = pd.cut(X, [-np.inf] + list(np.sort(coef)) + [np.inf],
labels = [0, 1, 2, 3])

        return -qwk(y, X_p)

    def fit(self, X, y):
        """
        Optimize rounding thresholds

        :param X: The raw predictions
        :param y: The ground truth labels
        """
        loss_partial = partial(self._kappa_loss, X=X, y=y)
        initial_coef = [0.5, 1.5, 2.5]
        self.coef_ = sp.optimize.minimize(loss_partial, initial_coef
, method='nelder-mead')

    def predict(self, X, coef):
        """
        Make predictions with specified thresholds

        :param X: The raw predictions
```

```
        :param coef: A list of coefficients that will be used for rou
nding
        """
        return pd.cut(X, [-np.inf] + list(np.sort(coef)) + [np.inf],
labels = [0, 1, 2, 3])


    def coefficients(self):
        """
        Return the optimized coefficients
        """
        return self.coef_['x']
```

In [14]:

```
%%time
pr1 = regressor_model1.predict(reduce_train)

optR = OptimizedRounder()
optR.fit(pr1.reshape(-1,), y)
coefficients = optR.coefficients()
```

```
CPU times: user 7.74 s, sys: 3.88 s, total: 11.6 s
Wall time: 7.73 s
```

In [15]:

```
opt_preds = optR.predict(pr1.reshape(-1, ), coefficients)
qwk(y, opt_preds)
```

Out[15]:

```
0.7083539663311538
```

In [16]:

```
# some coefficients calculated by me.
pr1 = regressor_model1.predict(reduce_test)
pr1[pr1 <= 1.12232214] = 0
pr1[np.where(np.logical_and(pr1 > 1.12232214, pr1 <= 1.73925866))] =
1
pr1[np.where(np.logical_and(pr1 > 1.73925866, pr1 <= 2.22506454))] =
2
pr1[pr1 > 2.22506454] = 3
```

In [17]:

```
sample_submission['accuracy_group'] = pr1.astype(int)
sample_submission.to_csv('submission.csv', index=False)
```

In [18]:

```
sample_submission['accuracy_group'].value_counts(normalize=True)
```

Out[18]:

Data

Data Sources

- 2019 Data Science Bowl
  - sample_submission.csv — 2 columns
  - specs.csv — 3 columns
  - test.csv — 11 columns
  - train.csv — 11 columns
  - train_labels.csv — 7 columns
- dsbowl19_features
  - generated_train_new.csv — 122 columns

2019 Data Science Bowl

Uncover the factors to help measure how young children learn

Last Updated: 2 months ago

About this Competition

In this dataset, you are provided with game analytics for the PBS KIDS *Measure Up!* app. In this app, children navigate a map and complete various levels, which may be activities, video clips, games, or assessments. Each assessment is designed to test a child's comprehension of a certain set of measurement-related skills. There are five assessments: Bird Measurer, Cart Balancer, Cauldron Filler, Chest Sorter, and Mushroom Sorter.

The intent of the competition is to use the gameplay data to forecast how many attempts a child will take to pass a given assessment (an incorrect answer is counted as an attempt). Each application install is represented by an `installation_id`. This will typically correspond to one child, but you should expect noise from issues such as shared devices. In the training set, you are provided the full history of gameplay data. In the test set, we have truncated the history after the start event of a single assessment, chosen randomly, for which you must predict the number of attempts. Note that the training set contains many `installation_id` s which never took assessments, whereas every `installation_id` in the test set made an attempt on at least one assessment.

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

Output Files

New Dataset New Notebook Download All

Output Files

- submission.csv

About this file

This file was created from a Kernel, it does not have a description.

submission.csv

1	installation_id	accuracy_group
2	00abaee7	3

3	01242218	3
4	017c5718	3
5	01a44906	3
6	01bc6cb6	2
7	02256298	3
8	0267757a	2
9	027e7ce5	2
10	02a29f99	0
11	0300c576	1
12	03885368	2
13	03ac279b	3
14	03e33699	3
15	048e7427	3
16	04a7bc3f	0
17	04d31500	0
18	0500e23b	2
19	0512bf0e	2
20	0525589b	3
21	05488e26	2
22	05771bba	2
23	05b82cf5	2
24	05e17e19	3
25	0617500d	3
26	068ae11f	1
27	0754f13b	0
28	07749e99	3
29	08611cc8	1
30	08671ec7	2
31	0889b0ae	3

Comments (76)

Sort by

All Comments

Hotness

Click here to comment...

kensler • Posted on Latest Version • 2 days ago • Options • Reply 1

Yours was the first Kaggle kernel I've ever forked! I imagine it wont be the last of yours. Thanks so much for helping me get started.

k_naga • Posted on Latest Version • 4 days ago • Options • Reply 1

Thanks for sharing!
Your Notebook was very useful to me.

Anatoly Zhukov • Posted on Latest Version • 5 days ago • Options • Reply 1

Thanks for sharing!

Subbu Vidyasekar • Posted on Latest Version • 5 days ago • Options • Reply 1

Superb Kernal Andrew.... understood the necessity of creating functions for data loading, encoding and more.. which means code reusability.

John Li • Posted on Latest Version • 6 days ago • Options • Reply 2

@artgor Thans for sharing your insightful kernal! When I ran this kernal on my personal server, an issue occurred: "LightGBMError: Do not support special JSON characters in feature name." And I had tried to reinstall different versions of 'lightgbm', could you please help me about this issus? Thanks for your precious time!

Andrew Lukyane... Kernel Author • Posted on Latest Version • 6 days ago • Options • Reply 1

It seems that for some reason column names have strange values in your case. You can try to look at them and find which column name causes the error.

John Li • Posted on Latest Version • 6 days ago • Options • Reply 1

Thank you, and I found the solution that fix my problem ---- changing kernel from python3.7 to python3.6. Maybe people like me can use conda to manage your multi-python envs. I think maybe some libs match to python3.7 makes the json files error.

Baligh Mnassri • Posted on Latest Version • 5 days ago • Options • Reply 0

Thanks for the answer. I have the same error that you. By checking the version of the installed python on Kaggle server, I found that is `Python 3.6.6 :: Anaconda, Inc`. What should I do in my case?

Osamu • Posted on Latest Version • 4 days ago • Options • Reply 1

I had the same error and changing 'title_count' column name from title name to title label number fixed this problem. I don't know exactly, but I saw somewhere that "," in column names causes this problem...

Chizuchizu • Posted on Latest Version • 2 days ago • Options • Reply 1

I had the same issue too. But I down the version of LGBM from 2.3.1 to 2.3.0 so it works normally.

If you want to use 2.3.1, you should read https://www.kaggle.com/c/data-science-bowl-2019/discussion/120344.

I'm sorry for bad English.

hisashi_h • Posted on Latest Version • 2 days ago • Options • Reply 0

I had the same error.
There seems to be a column that contains a comma like "f*Heavy,*Heavier,_Heaviest".
It was solved by putting the following sentence.

```
reduce_train.columns = reduce_train.columns.str.replace(',', '')
```

Improving_Renyimi... • Posted on Latest Version • 8 days ago • Options • Reply 1

Thanks for sharing!👍 👍 👍

Aspgvu • Posted on Latest Version • 9 days ago • Options • Reply 1

Is there a way to save the trained model? (regressor_model1) The unpickled object produce different result for some reason. Thanks for sharing!

Andrew Lukyane... Kernel Author • Posted on Latest Version • 6 days ago • Options • Reply 0

Hm, strange. I thought that pickling the class should be stable.

fingul • Posted on Latest Version • 10 days ago • Options • Reply 1

Thanks for sharing!👍

Unknown Error • Posted on Latest Version • 11 days ago • Options • Reply 1

Thanks for sharing!👍

Akira Sato • Posted on Latest Version • 12 days ago • Options • Reply 1

Thank you for sharing your insight! Extremely helpful! :)

Wei Hao Khoong • Posted on Latest Version • 14 days ago • Options • Reply 1

Awesome kernel! Thanks for sharing :)

Ma'alona Mafaufau • Posted on Latest Version • 17 days ago • Options • Reply 1

Thank you for uploading such an insightful kernel @artgor. As my first serious attempt at a machine learning competition, this has been extremely helpful in getting started. All the best to you :)

Mikhail Sokolov • Posted on Latest Version • 25 days ago • Options • Reply 7

Hi @artgor !Thank you for sharing your ideas.
One thing that doesn't give me rest is the fact that **preprocess** function gets **reduce*train **and **reduce*test **which are different by structure. Eventually, after preprocessing, additional statistics, added as new columns, calculates differently and I wonder how it comes that the final score is so high.

What I mean here is that reduce*train has all assessments' history while reduce*test has only the last ones. Consequently, the calculation of **mean**, **nunique**, **count ** for the train set is based on different input data compared to the test set. As you could see, for example, '**installation*session*count**' column has values '1' for all observations in the output reduce*test dataset and the '**installation**duration*mean' value always equals the '**duration**mean'. That means that we produce columns either those duplicate existing columns or have zero variance.

I tried to fix it and recreated reduce*test the way it looked like reduce*train, i.e. with all assessment history inside. After applying the preprocessing function and then trained the model and predicting the scores for submission, I got a plunge in LB score to .40

So, just curious why this strange preprocessing works better than, as it would seem, the correct one.

Andrew Lukyane... Kernel Author • Posted on Latest Version • 24 days ago • Options • Reply 1

To tell the truth, I don't understand this myself. As I have already written, I took all the features from that kernel without changing.
Sometimes on Kaggle strange ideas work... though sometimes they cause severe overfitting - and in this competition public part is quite small.
So I'm not sure that it is worth using such features.

Shuxun Zhang • Posted on Latest Version • 22 days ago • Options • Reply 0

Same problem. I droped all the features in preprocess and added some more which should be better, but unfortunately, the LB socre got a plunge too.

boydbigdatarpg • Posted on Latest Version • 23 days ago • Options • Reply 3

Very powerful technique by wrapping all train and test preprocessing in the function
and I can learn a lot from pandas methods
Thank you very much for sharing =] @artgor 👍
Cheers

Y. O. • Posted on Latest Version • 21 days ago • Options • Reply 1

Thank you @artgor! Do you have any idea about the bump on the learning curve around 20 iterations?

YaGana Sheriff-Hus... • Posted on Latest Version • 25 days ago • Options • Reply 3

@artgo, thanks for sharing. I forked and ran your kernel as is but submitting it to the public LB scored 0.525

So it has the same reproducibility issue as the LGB and catboost kernels. With the tree methods I thought it is a CV issue but did not expect it from a linear model. Does anybody know why this happens?

Andrew Lukyane... Kernel Author • Posted on Latest Version • 25 days ago • Options • Reply 2

This kernel uses lgb regression model, so it is also a gradient boosting. To fix randomness it is necessary to define all random seeds in the model parameters.

YaGana Sheriff-... • Posted on Latest Version • 25 days ago • Options • Reply 2

Thanks @artgo. I did fix all seeds in my LGB and Catboost kernels but still had reproducibility issues. When I submit the same result twice, I get different scores in all cases including a fork of this kernel that got LB=0.525 and LB=0.520

S. Panchenko • Posted on Latest Version • 22 days ago • Options • Reply 1

There are some np.random in data preprocessing (class MainTransformer(BaseEstimator, TransformerMixin)), so you need to add np.random.seed(10), random.seed(10)

Srinath Jayachan... • Posted on Latest Version • 16 days ago • Options • Reply 0

@sheriytm , wondering if you were able to solve the lightgbm's reproducibility issue?

YaGana Sheriff-... • Posted on Latest Version • 15 days ago • Options • Reply 0

@pepslee, as I mentioned above, I have set all the seeds but did not make a difference as per reproducibility.

@srinath9s, no but I have been busy with another competition. Since fixing all random seeds did not take care of result reproducibility issue, I have a feeling it has something to do with the data and I have not had time to look into that yet. Do let me know if you are able to do it before me.

S. Panchenko • Posted on Latest Version • 9 days ago • Options • Reply 0

@sheriytm have you look into data preparation part ? it seems there some random in data loading or preprocessing

Maksim Rodin • Posted on Latest Version • 25 days ago • Options • Reply 3

Hi @artgor,
I actually wonder, is it not some kind of recursion with the OptimizedRounder coefficient?
Cause you use the coefficients in the evaluation function `eval_qwk_lgb_regr` and train the model with early stopping, which is based on this evaluation function.
And then you use trained model to find these coefficients?
Am I correct here or do I miss something?

Regards

Andrew Lukyane... Kernel Author • Posted on Latest Version • 25 days ago • Options • Reply 3

A good point for a discussion.

I'd say there are multiple possible approaches:

- use default values for metric calculation while training;
- find optimal values at each iteration (but it will be slow) on train/valid set;
- find optimal values for metric on each fold;
- find optimal values for the oof predictions;
- other ideas;

I think it is necessary to check all of them and find which works best.

Buffalo Spdwy • Posted on Latest Version • 20 days ago • Options • Reply 2

@artgor thanks for the kernel! @bi0max thanks for this discussion.
Big Big thanks to @braquino for the feature ideas!

... can't stop thanking, too many @.

I have a question here.

Reading this interesting kernel, I feel, one aspect of the regression being better than the classification approach is that it incorporates the fact that the accuracy group is naturally ordered. The "0,1,2,3" accuracy group is not like "dog, car, bench, tree" but like "bad, normal, good, great". Regression is naturally ordered.

But, "bad, normal, good, great" is not necessarily "0,1,2,3" numerically in how good they are. How they are encoded, e.g. perhaps change "0,1,2,3" to "-10,1,2,3" to indicate accuracy group 0 has a long tail of bad performance, would effect the final classification. By changing encoding, perhaps the regression prediction strength rank of some test samples would also change, which means given two different encoding, there do *NOT* necessarily exist two sets of thresholds that would make the two encoding end up with the same classification result.

So, would the best be optimize the accuracy group encoding and the threshold together? 😃

The encoding optimization would involve refitting the model though.

Bruno Aquino • Posted on Latest Version • 19 days ago • Options • Reply 3

Hi @barnwellguy , the output of a Gradient Booster regressor isn't linear so, you don't need to care about the linearity of the accuracy groups, beyond that, a non linear round is applied in the end.

Buffalo Spdwy • Posted on Latest Version • 19 days ago • Options • Reply 2

Hi @braquino , thanks for replying!

When you say "the output of a Gradient Booster regressor isn't linear", what is the linearity with respect to?

- In prediction, the gradient boost regressor as a function of one sample's feature vector to one accuracy score is nonlinear, since it is a tree-based method.
- In the bigger picture, the gradient boost regressor as a function from a training objective encoding to its prediction on any specific sample is much harder to say. For example, I think, principally, using labels "0,10,20,30" will give you predictions exactly 10 times as big as predictions given by using labels "0,1,2,3", since GBM regressor is also bin average based. This is interesting. Given a specific encoding vector

  $$E = [a, b, c, d]$$

  if we use function $G(E, \text{sample})$ to denote the the output of a regressor trained using encoding E predicting a specific sample. Then, with any non-zero real number θ, we define a partial function

  $$F(\theta) := G(\theta E, \text{sample})$$

  I think F is actually **linear**, in the sense that, for any real numbers α, β, with any non-zero real number θ, γ, that satisfies $\alpha\theta + \beta\gamma \neq 0$,

  $$F(\alpha\theta + \beta\gamma) = \alpha F(\theta) + \beta F(\gamma)$$

  Of course, this is far from the $G(E_1, \text{sample}) + G(E_2, \text{sample}) = G(E_1 + E_2, \text{sample})$ type of linear, when vectors E_1, E_2 are not linearly dependent.

But forget about the linearity we talk about above. I deviated. Linearity of any kind is perhaps not very to the point of this discussion.

Here, I think the loss of the regression matters.

For example, if we change the objective from "0, 1, 2, 3" to "0, 0.01, 2, 2.01", then, I believe, the regressor and the subsequently thresholded classifier should perhaps perform much worse in differentiating the first-second and third-fourth class, since in the regression, there is almost no loss that motivate the fitter to differentiate the first-second and third-fourth classes.

Thanks for reading, what do you think?

Massoud Hosseinali • Posted on Latest Version • 25 days ago • Options • Reply 3

Thanks for sharing this Andrew.
It's worth giving credit to @yasufuminakama who did the regression part in this competition first at https://www.kaggle.com/yasufuminakama/public-dsb2019-lgbm-regression-sample but it didn't get the attention it deserved.

Andrew Lukyane... Kernel Author • Posted on Latest Version • 25 days ago • Options • Reply 0

Thanks for mentioning that kernel - I have completely missed it.

DarkCat • Posted on Latest Version • 24 days ago • Options • Reply 2

you are a true gent for sharing this insight. this is by a long margin the clearest single light source so far into this comp. you may have lost 100k w one click, but now you have some fans! I too have been fiddling with regressions but kept getting stuck on how to divvy up the spoils. brilliant! thanks!

olaf • Posted on Latest Version • 25 days ago • Options • Reply 2

Great work! Only... This competition now became a lottery - the difference between this score and the best one is less than one standard deviation of the error introduced by public test set sampling - see https://www.kaggle.com/olafmat/leaderboard-score-confidence-interval
So in a moment there will be 1000 people with slightly different versions of this script, each one with quite a chance to reach the first place.

Andrew Lukyane... Kernel Author • Posted on Latest Version • 25 days ago • Options • Reply 4

There are two months more till the end of the competitions. I'm sure there will be multiple kernels with higher score and the score of my kernel will drop to bronze or lower.

olaf • Posted on Latest Version • 24 days ago • Options • Reply 1

Hopefully minor changes are going to be highly correlated, so either they stand together on the podium or fall together. Certainly this notebook is great help for beginners, which should be appreciated!

Bruno Aquino • Posted on Latest Version • 23 days ago • Options • Reply 1

Thank you @artgor , your idea helped me a lot, I bet that all top scores use regression.

MoHenni • Posted on Latest Version • 23 days ago • Options • Reply 0

So, if by chance someone who used your kernel gets first place on the private leaderboard does he or she ows you half of the price?

Andrew Lukyane... Kernel Author • Posted on Latest Version • 22 days ago • Options • Reply 4

It will be a valueable/expensive lesson to me in that case :)

Mashlyn • Posted on Latest Version • 25 days ago • Options • Reply 1

Nice kernel!!! Thanks for sharing!😄 👍

Aldrin • Posted on Latest Version • 25 days ago • Options • Reply 1

Cool regression approach! Thanks for sharing.

LeiZhou • Posted on Latest Version • a month ago • Options • Reply 1

Thanks!!
I don't quite understand how the coef optimizer part work. How and To what extent nelder-mead method optimize the loss?

Andrew Lukyane... Kernel Author • Posted on Latest Version • a month ago • Options • Reply 1

It simply finds the best thresholds to optimize the score.

ragnar • Posted on Latest Version • a month ago • Options • Reply 1

Nice aproach!!. Thanks!!.

Manoj Prabhakar • Posted on Latest Version • a month ago • Options • Reply 1

Great kernel @artgor ... Its actually amazing that regression worked for this task and it gave good LB score. Thanks a lot for sharing this approach. This is the first time I am seeing this for tabular data competition.

fingul • Posted on Latest Version • a month ago • Options • Reply 1

hmm... good

Neelam • Posted on Latest Version • a month ago • Options • Reply 0

Hi @manojprabhaakr

Do you want to make team with me?

e-toppo • Posted on Latest Version • 25 days ago • Options • Reply 2

@artgor Thanks for sharing.
How to calculate coefficients?(1.12232214, 1.73925866, 2.22506454)
Why you don't use coefficients calculated by nelder-mead method for submit file?

Andrew Lukyane... Kernel Author • Posted on Latest Version • 25 days ago • Options • Reply 2

I calculated them while developing this kernel (with OptimizedRounder) and used them.

YujiAriyasu • Posted on Latest Version • 25 days ago • Options • Reply 2

Thank you!
Can we use accuracy instead of accuracy*group and convert accuracy to accuracy*group at the end?
has no effect?
I'm a beginner, so let me know what you think.

Andrew Lukyane... Kernel Author • Posted on Latest Version • 25 days ago • Options • Reply 0

We can do it, but we don't be able to caclulate accuracy for the new assessments for which we need to make predictions.

HawkWang • Posted on Latest Version • 22 days ago • Options • Reply 0

may be we can predict accuracy, instead of accuracy group. And then we convert accuracy to accuracy group. Here we know accuracy is a float number, so it is naturally a regression issue.

Andrew Lukyane... Kernel Author • Posted on Latest Version • 22 days ago • Options • Reply 0

Hm. Maybe it could work.

Erik Bruin • Posted on Latest Version • a month ago • Options • Reply 2

Great! Yes, it was also used in PetFinder for instance.

Manoj Prabhakar • Posted on Latest Version • 25 days ago • Options • Reply 0

Thanks for sharing it @erikbruin .. I was not active during that time. My bad, I would have missed it. Good to learn.

Christina Morgan • Posted on Latest Version • 11 hours ago • Options • Reply 0

Thank you for sharing!

Dalek • Posted on Latest Version • 5 days ago • Options • Reply 0

How do you choose these thresholds?

Mahmoud farfoura • Posted on Latest Version • 5 days ago • Options • Reply 0

Dear @artgor,
Thankx for sharing the nice kernel, i am curious some installation_is's don't have rows in training data file? did you notice this??
Does this affect our scores as well?
Appreciated.

Andrew Lukyane... Kernel Author • Posted on Latest Version • 5 days ago • Options • Reply 0

I didn't use them for training.

kavi • Posted on Latest Version • 6 days ago • Options • Reply 0

@artgor Thanks for sharing your kernel,i forked your kernel and changed the feature engineering part and extracting json data as used in this kernel https://www.kaggle.com/pestipeti/memory-efficient-faster-way-to-extract-json-data . I got CV score on train: 0.7835 +/- 0.0066 std and CV score on valid: 0.7788 +/- 0.0182 std. I also got qwk score as 1.0
but my LB score is -0.010. can you please guide me in this i can't understand why my LB is very low even cv and qwk score are good.

Andrew Lukyane... Kernel Author • Posted on Latest Version • 6 days ago • Options • Reply 0

I suppose this score means that you are either predicting the same class for most of the samples or your features for train and test data are different (or in a different order), so predictions are wrong.

fanoping • Posted on Latest Version • 11 days ago • Options • Reply 0

Thanks for sharing! Impressive work!

Hamza • Posted on Latest Version • 11 days ago • Options • Reply 0

Hi @artgor nice kernel!
Can you please explain me how you calculated those values to differentiate the outputs??

1.12232214
1.73925866

Erik Bruin • Posted on Latest Version • 12 days ago • Options • Reply 0

Hi Andrew, Just a quick question. I saw that you inserted boundaries manually in the final rounding, but shouldn't those just be the ones in the coefficients list (I get some slightly different numbers if I run your kernel)?

Andrew Lukyane... Kernel Author • Posted on Latest Version • 12 days ago • Options • Reply 1

I agree that it would be better to use optimized coefficients. I used the coefficients from my previous run because I forgot to change them to optimized.

Erik Bruin • Posted on Latest Version • 11 days ago • Options • Reply 0

Ok, I already thought something like that. Thanks.

MhdSharuk • Posted on Latest Version • 24 days ago • Options • Reply 0

I don't understand why did he used **argmax(axis=0)** function in **eval qwk lgb regr()**

jajaja • Posted on Latest Version • 25 days ago • Options • Reply 0 thx! You've given me a lot of insight!👍

6 days ago

This Comment was deleted.

11 days ago

This Comment was deleted.

Showing 50 of 76 comments. Show More

Similar Kernels

Our Team Terms Privacy Contact/Support