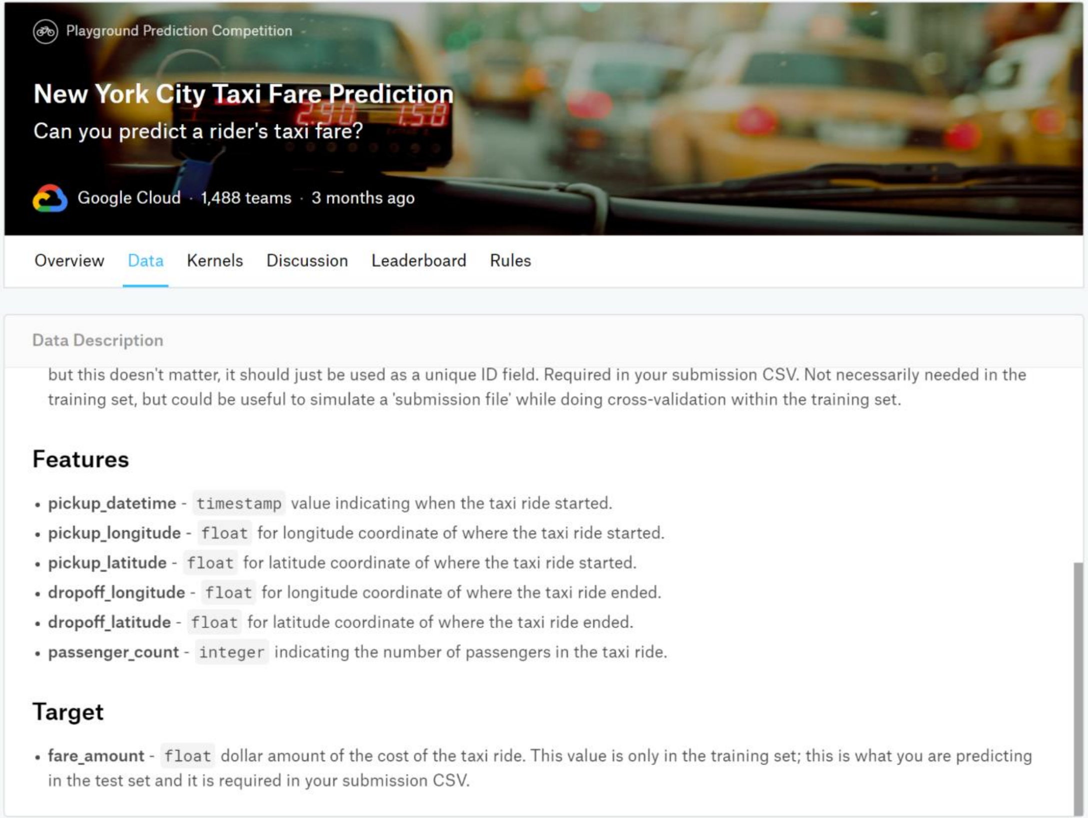
New-york-city-taxi-fare-prediction

組員:

- 會延A 03152138 張永霖
- 財經四A 04155136 馮顥典
- 巨資四A 04170104 游惠如
- 巨資四A 04170116 沈芳儀
- 巨資四A 04170118 洪聆紘
- 巨資四A 04170133 王士豪

new-york-city-taxi-fare-prediction

https://www.kaggle.com/c/new-york-city-taxi-fare-prediction/data



Playground Prediction Competition

New York City Taxi Fare Prediction

Can you predict a rider's taxi fare?

Google Cloud · 1,488 teams · 3 months ago

Overview Data Kernels Discussion Leaderboard Rules

Data Description

but this doesn't matter, it should just be used as a unique ID field. Required in your submission CSV. Not necessarily needed in the training set, but could be useful to simulate a 'submission file' while doing cross-validation within the training set.

Features

- **pickup_datetime** - `timestamp` value indicating when the taxi ride started.
- **pickup_longitude** - `float` for longitude coordinate of where the taxi ride started.
- **pickup_latitude** - `float` for latitude coordinate of where the taxi ride started.
- **dropoff_longitude** - `float` for longitude coordinate of where the taxi ride ended.
- **dropoff_latitude** - `float` for latitude coordinate of where the taxi ride ended.
- **passenger_count** - `integer` indicating the number of passengers in the taxi ride.

Target

- **fare_amount** - `float` dollar amount of the cost of the taxi ride. This value is only in the training set; this is what you are predicting in the test set and it is required in your submission CSV.

增加的額外變數

解釋增加變數

01 **年**:將pick_datatime中的year分出。

02 **月**:將pickup_datetime中的month分出。

03 日:將pickup_datetime中的date分出。

04 小時:將pickup_datetime中的hour分出。

05 星期:透過「年、月、日」得出對應星期

解釋增加變數

06 **時間區段**: 以小時區分

07 經緯距離(有算弧度)**:**根據上下車地點的經緯度計算距離。(單位為KM)

08 **經緯距離(直線距離):**根據上下車地點的經緯度計算距離。(單位為KM)

09 距離級距: 用NO.7的來做區段依據

10 上車地點: 由上車地點經緯度得出

解釋增加變數

11 **下車地點:**由下車地點的經緯度得出。

12 行進方向:透過上下車經緯度區分。

13 **跨區與否:**上下車地點是否為紐約市

14 移動跨區與否:ex.市區→市內、市區內移動……等。

套件使用

- **Numpy**
- **Pandas**
- **Time**
- **Sys**
- **Matplotlib.pyplot**
- **Saeborn**
- **RandomForestRegressor**
- **Reverse_geocoderPandas**
- **Sklearn.model_selection**
- **Lightgbm**
- **Holiday**
- **Sklearn**
- **CatBoostRegressor**

刪除含異常值之資料

- 經度小於 -90或大於0
- 緯度小於0或大於90
- 乘客人數小於1或大於7
- 價格小於2.5或大於200
- 距離為0或大於小於三個標準差

RandomForestRegressor
&
GradientBoostingRegressor

RandomForestRegressor

使用套件：
- Numpy
- pandas
- os
- time
- sys
- matplotlib.pyplot
- saeborn
- RandomForestRegressor

使用資料：
Train.csv之前1000000筆資料

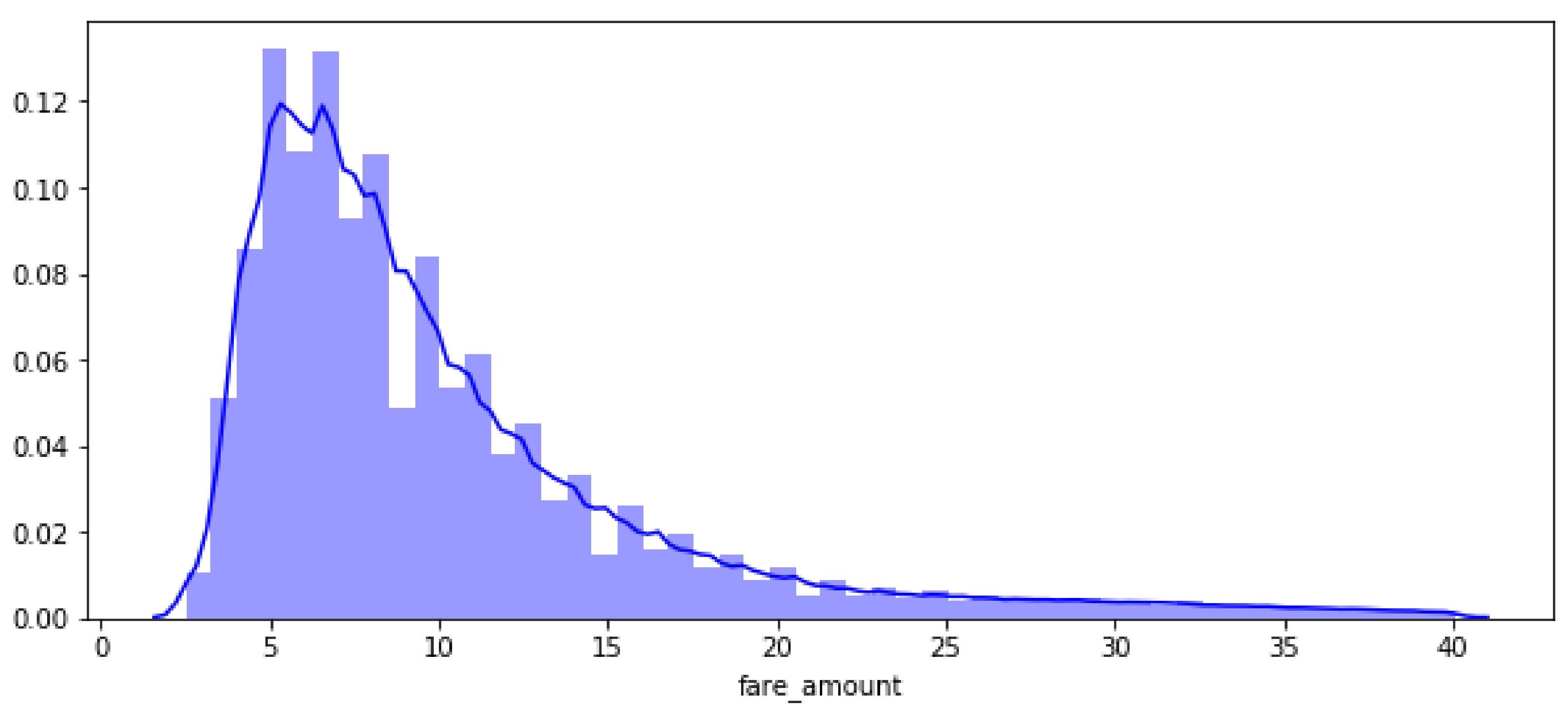
初步刪除空值、異常值
- 經度小於 -90或大於0
- 緯度小於0或大於90
- 乘客人數小於0或大於7
- 價格小於2.5或大於200

增加了十九項變數

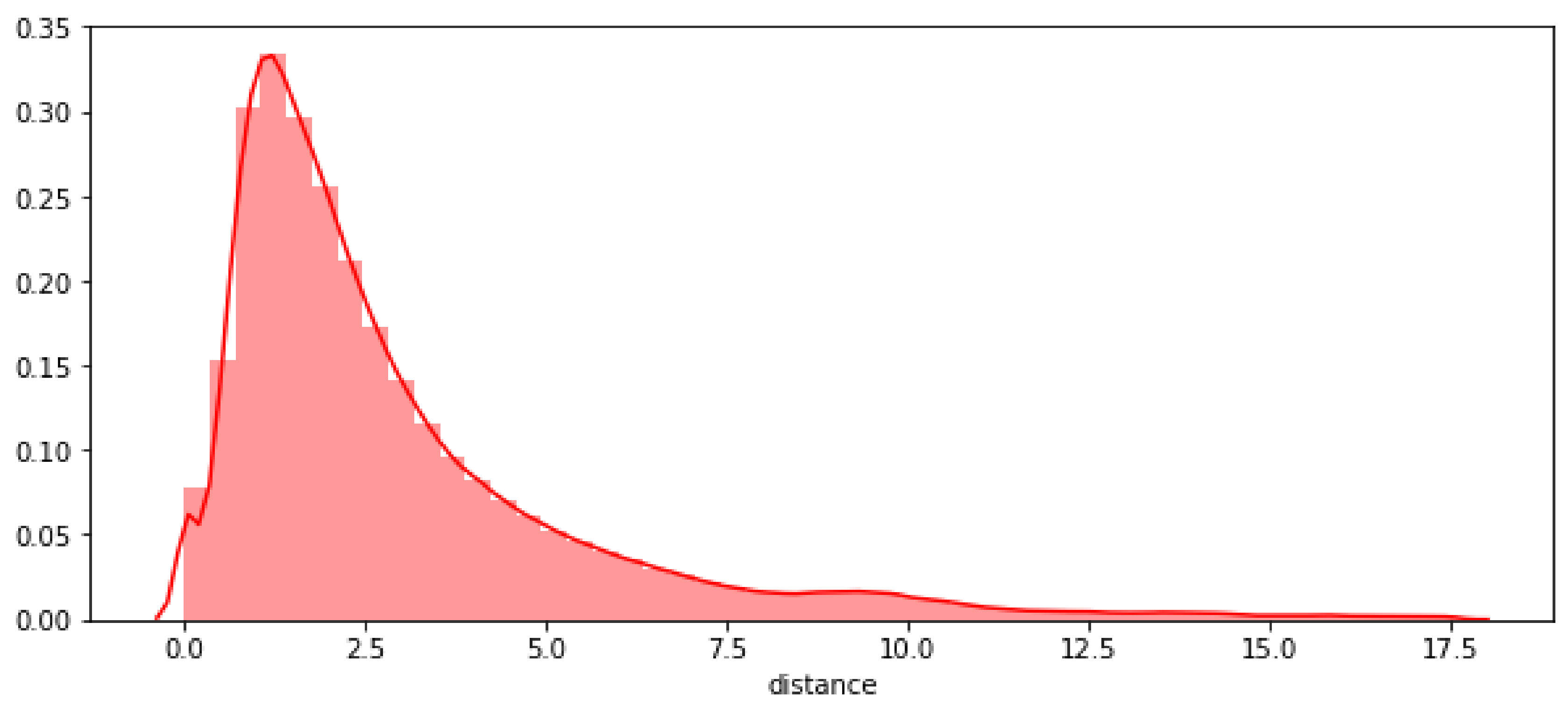
- distance（刪除距離為0值、距離大於三個標準差）
- distance_grade（依四分位數切割）
- pickup_place、dropoff_place（依據經緯度得出所在地區）
- pickup_place2、dropoff_place2（是否為紐約市與其五大行政區）
- area_across（市內：紐約市及其行政區；市外：其他）

  市外移動 / 市內移動 / 市內到市外 / 市外到市內
- direction（移動方向：NW、NE、SW、SE)
- direction轉換之dummy變數
- direction_num（由小到大依各方位平均費用賦值）
- year 、 month 、 date、 weekday、 hour
- time_interval（依「小時」直方圖切割四等分）

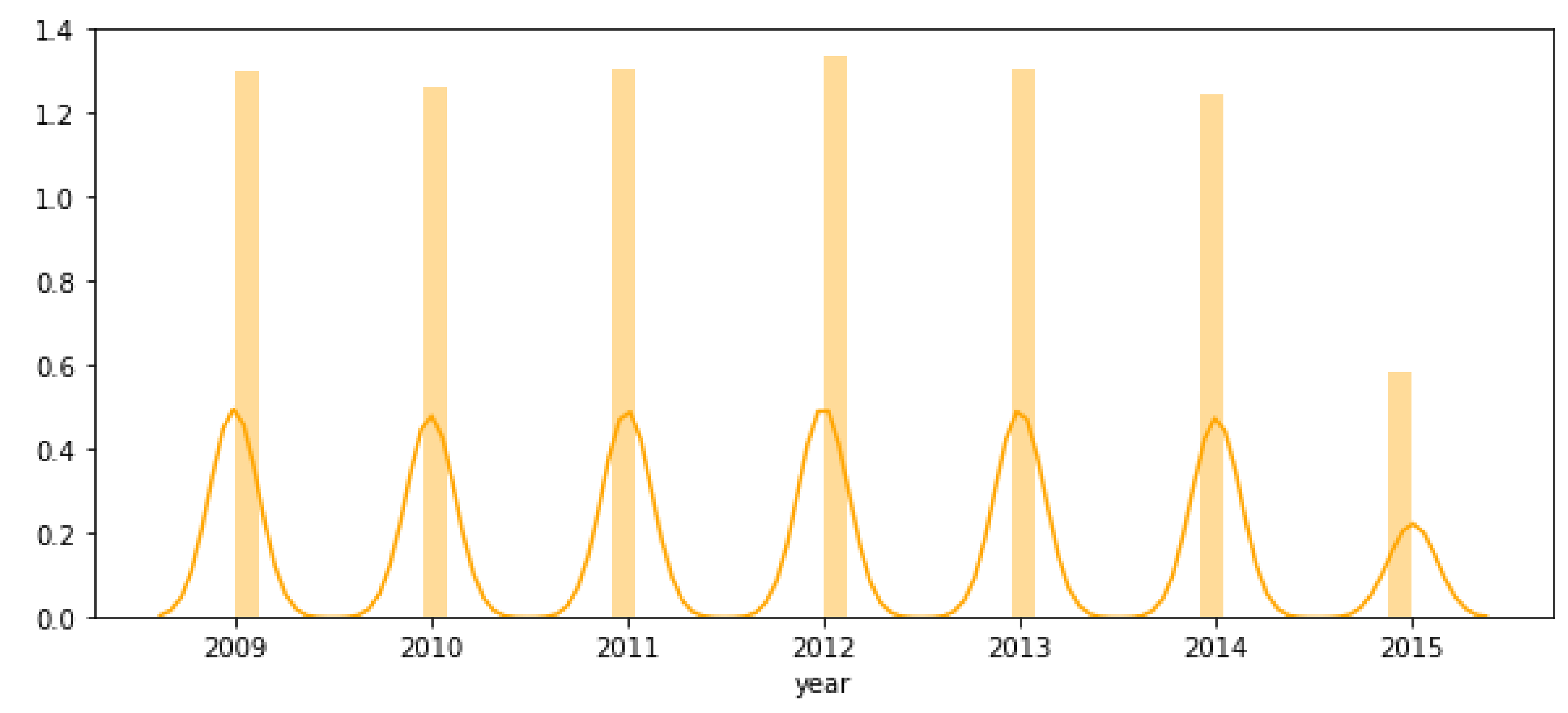
價格分布(小於mean+3*std)



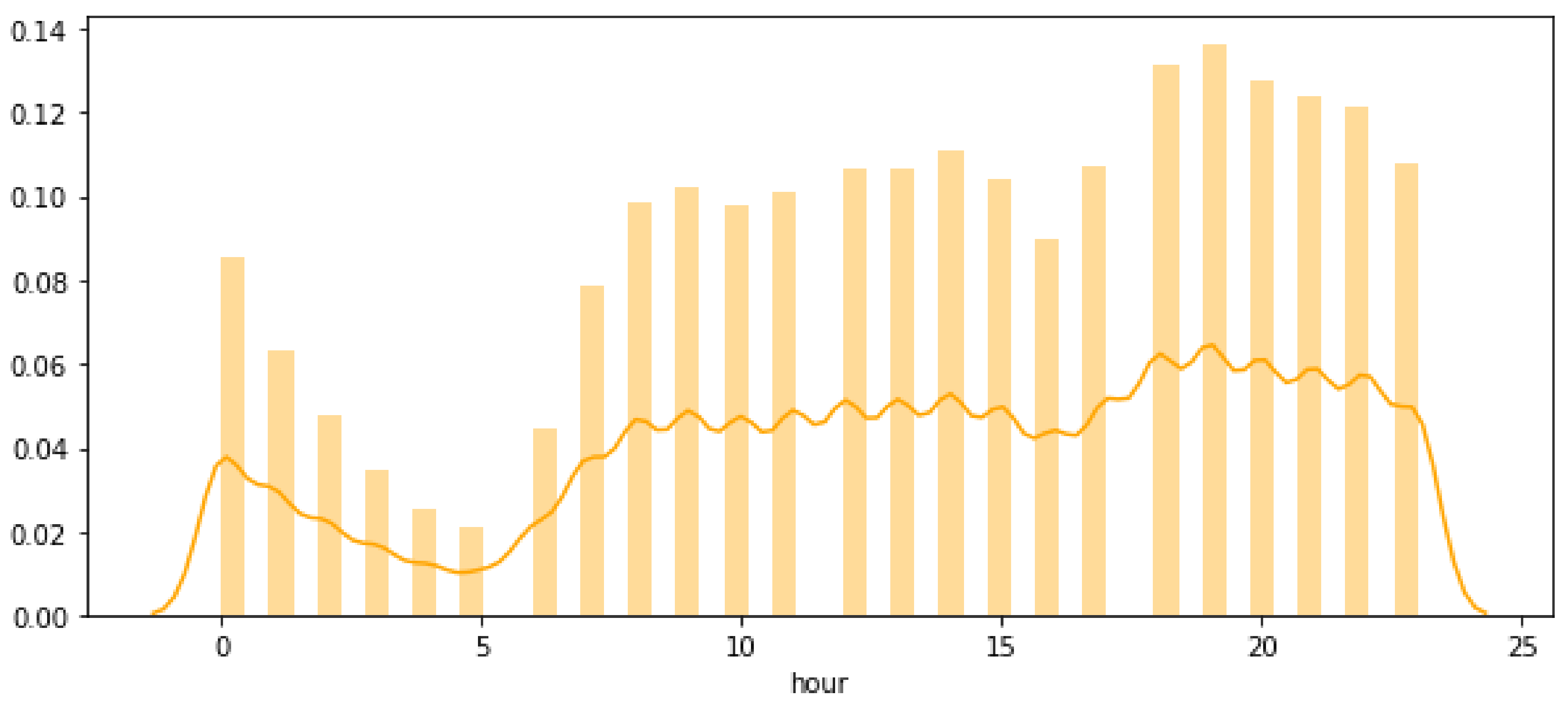
距離分布(小於mean+3*std)



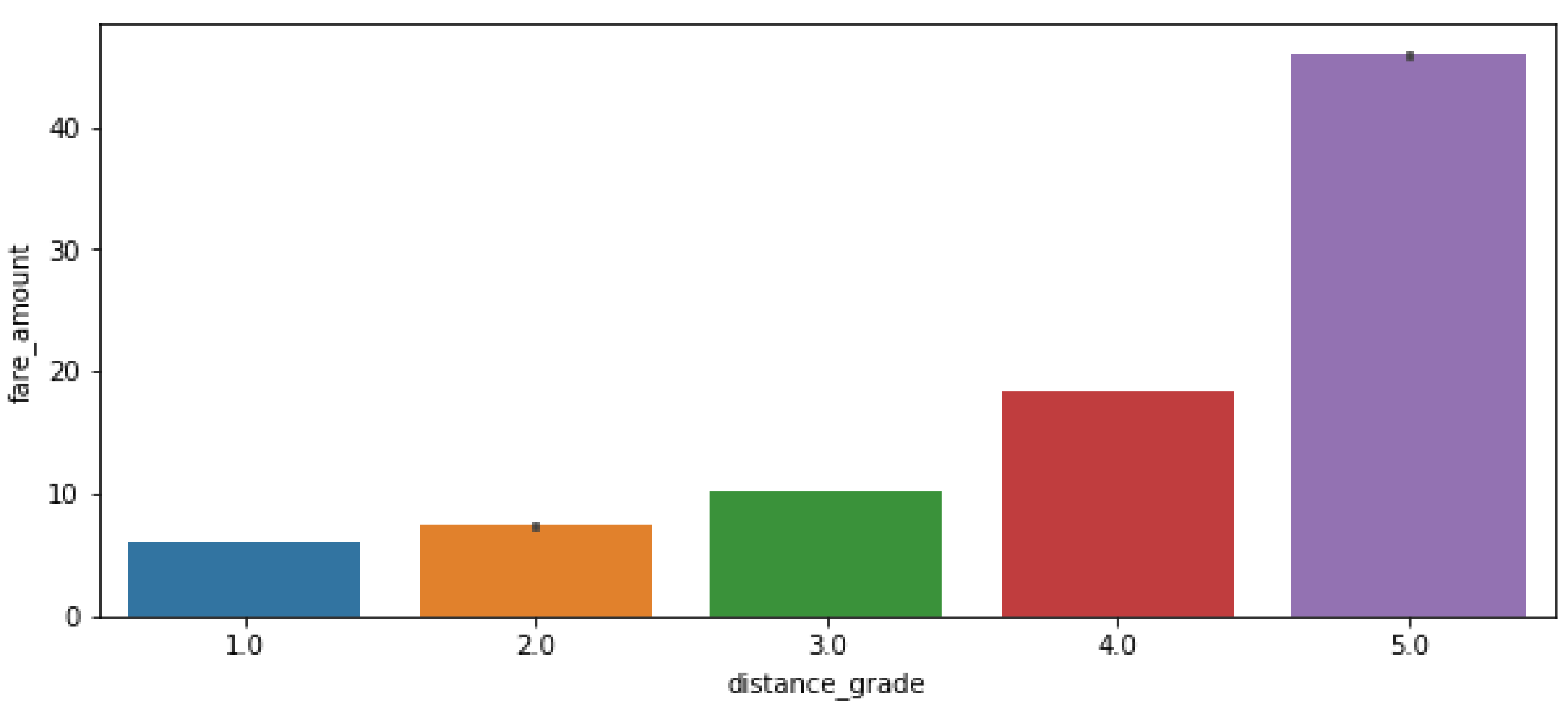
年份分布



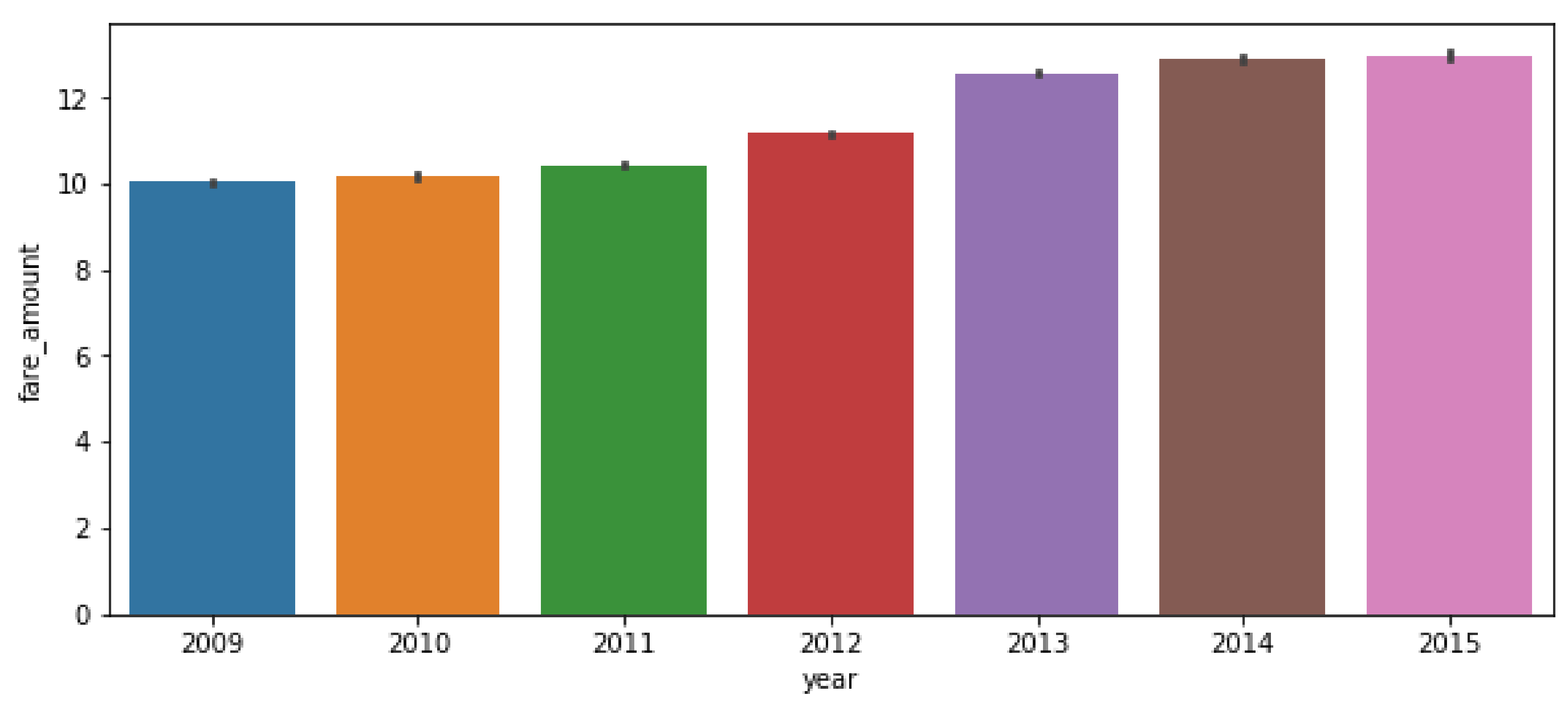
時數分布



距離與其價格分布

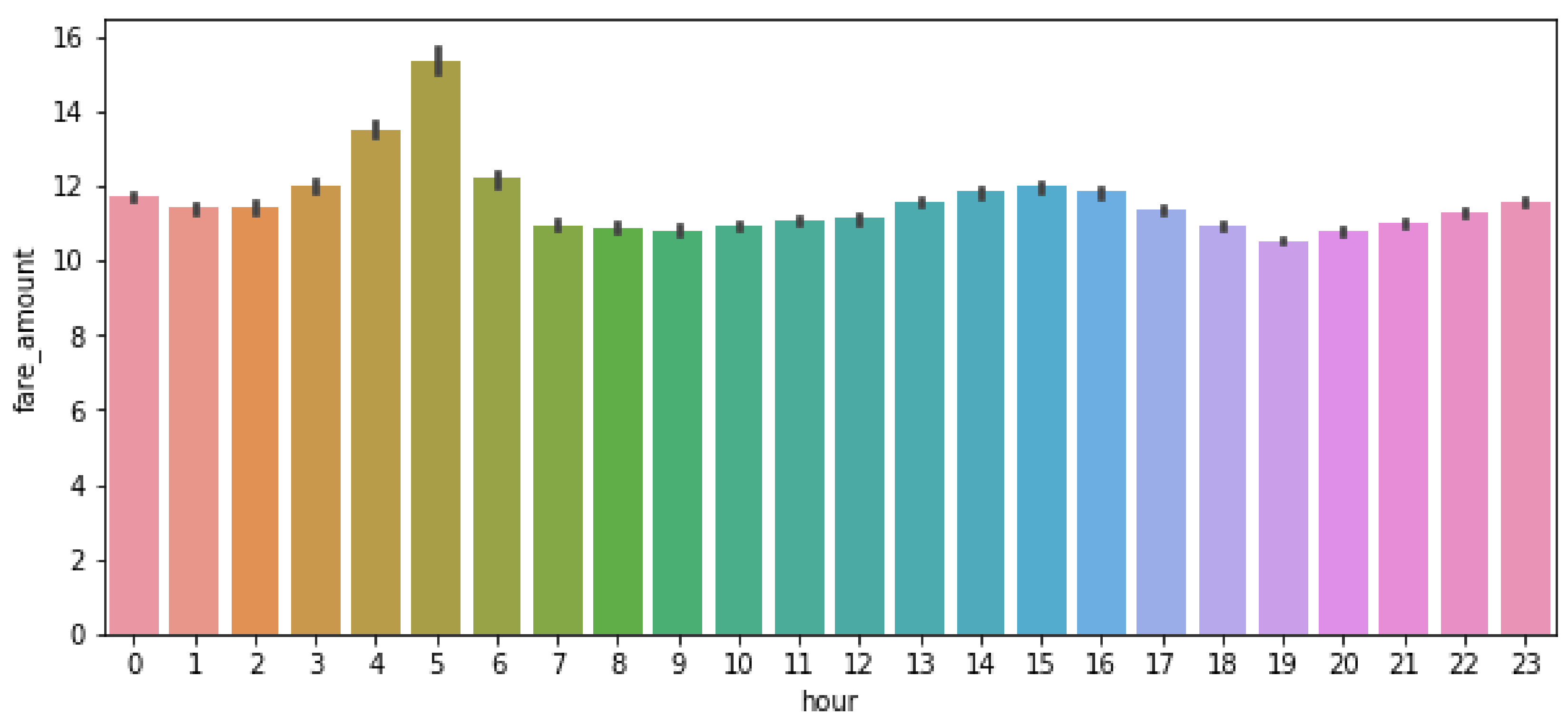


年份與其價格分布



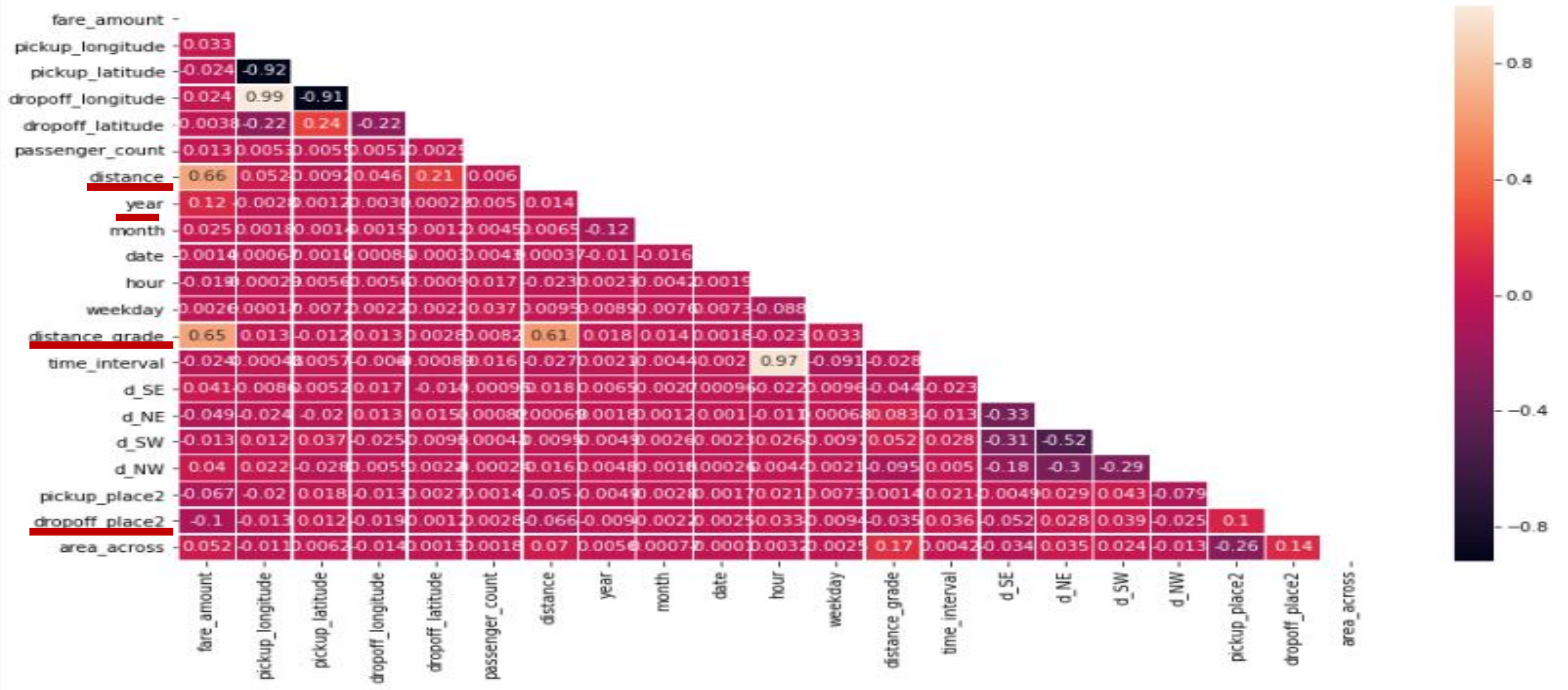
fare_amount
0
2
4
6
8
10
12
2009
2010
2011
2012
2013
2014
2015
year

小時與其價格分布



16
14
12
10
8
6
4
2
0
fare_amount
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
hour

欄位間相關係數


fare_amount
pickup_longitude
pickup_latitude
dropoff_longitude
dropoff_latitude
passenger_count
distance
year
month
date
hour
weekday
distance_grade
time_interval
d_SE
d_NE
d_SW
d_NW
pickup_place2
dropoff_place2
area_across
0.8
0.4
0.0
−0.4
−0.8

RandomForestRegressor

```
x = train_2.drop(["key","fare_amount","direction","pickup_place","dropoff_place","d_SE","d_NE"
,"d_SW","d_NW"],axis=1)
y = train_2['fare_amount'].values

# 從train中分割训练数据和测试数据
Num_train = int(x.shape[0]*0.8)
x_train, y_train = x[:Num_train], y[:Num_train]
x_test, y_test = x[Num_train:], y[Num_train:]

rf = RandomForestRegressor()
rf.fit(x_train, y_train)
rf_y_predict = rf.predict(x_test)

print("accuracy:", rf.score(x_test, y_test))
```

```
accuracy: 0.8331315193735873
```

```
x_train = train_2.drop(["key","fare_amount","direction","pickup_place","dropoff_place","d_SE",
"d_NE","d_SW","d_NW"],axis=1)
y_train = train_2['fare_amount'].values
x_test = test_2.drop(["key","direction","pickup_place","dropoff_place","d_SE","d_NE","d_SW","d
_NW"],axis=1)


rf = RandomForestRegressor()
rf.fit(x_train, y_train)
submission = pd.read_csv("../input/new-york-city-taxi-fare-prediction/sample_submission.csv")
submission['fare_amount'] = rf.predict(x_test)
print(submission.head(10))
submission.to_csv('submission_3.csv', index=False)
```

Score
3.37503

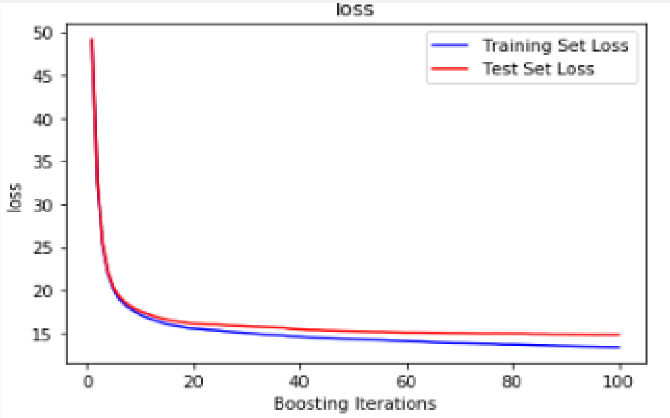
GradientBoostingRegressor

```python
x = train_2.drop(["key","fare_amount","direction","pickup_place","dropoff_place","d_SE","d_NE"
,"d_SW","d_NW"],axis=1)
y = train_2['fare_amount'].values

# 從train中分割訓練數據和測試數據
Num_train = int(x.shape[0]*0.8)
x_train, y_train = x[:Num_train], y[:Num_train]
x_test, y_test = x[Num_train:], y[Num_train:]

i=0.1
while i <1:
    gbc = GradientBoostingRegressor(learning_rate=i)
    gbc.fit(x_train, y_train)
    gbc_y_predict = gbc.predict(x_test)
    mse = mean_squared_error(y_test,gbc.predict(x_test))
    print("learning rate=",i)
    print("MSE: %.4f" % mse)
    print("accuracy: ", gbc.score(x_test, y_test))
    i+=0.1
```

```
learning rate= 0.1
MSE: 15.7788
accuracy:  0.8291008870603802
learning rate= 0.2
MSE: 15.1555
accuracy:  0.8358505914829565
learning rate= 0.30000000000000
MSE: 14.8569
accuracy:  0.8390847125865964
learning rate= 0.4
MSE: 14.7756
accuracy:  0.8399659551232508
learning rate= 0.5
MSE: 14.9630
accuracy:  0.8379361356789641
```

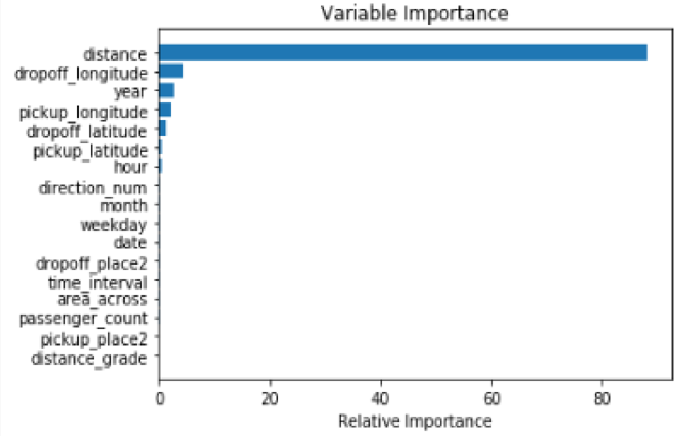
Loss

```python
test_score = np.zeros((100,), dtype=np.float64)

for i, y_pred in enumerate(gbc.staged_predict(x_test)):
    test_score[i] = gbc.loss_(y_test, y_pred)

plt.title('loss')
plt.plot(np.arange(100) + 1, gbc.train_score_, 'b-',
         label='Training Set Loss')
plt.plot(np.arange(100) + 1, test_score, 'r-',
         label='Test Set Loss')
plt.legend(loc='upper right')
plt.xlabel('Boosting Iterations')
plt.ylabel('loss')
```



Important features

```python
train_2_drop= train_2.drop(["key","fare_amount","direction","pickup_place","dropoff_place","d_SE","d_NE","d_SW","d_NW"],axis=1)
#畫圖-2 #feature importance
feature_importance = gbc.feature_importances_
feature_importance = 100.0 * feature_importance
sorted_idx = np.argsort(feature_importance)
pos = np.arange(sorted_idx.shape[0]) + .5
plt.barh(pos, feature_importance[sorted_idx], align='center')
plt.yticks(pos, train_2_drop.columns.values[sorted_idx])
plt.xlabel('Relative Importance')
plt.title('Variable Importance')
plt.show()
```



GradientBoostingRegressor –預測(1)

```
x_train = train_2.drop(["key","fare_amount","direction","pickup_place","dropoff_place","d_SE",
"d_NE","d_SW","d_NW"],axis=1)
y_train = train_2['fare_amount'].values
x_test = test_2.drop(["key","direction","pickup_place","dropoff_place","d_SE","d_NE","d_SW","d
_NW"],axis=1)

gbc = GradientBoostingRegressor(learning_rate=0.4)
gbc.fit(x_train, y_train)
submission = pd.read_csv("../input/new-york-city-taxi-fare-prediction/sample_submission.csv")
submission['fare_amount'] = gbc.predict(x_test)
print(submission.head(10))
submission.to_csv('submission_5.csv', index=False)
```

Score
3.33453

GradientBoostingRegressor –預測(2)

```
gbc = GradientBoostingRegressor(learning_rate=0.4,subsample=0.8,max_depth=10)
gbc.fit(x_train, y_train)
submission = pd.read_csv("../input/new-york-city-taxi-fare-prediction/sample_submission.csv")
submission['fare_amount'] = gbc.predict(x_test)
print(submission.head(10))
submission.to_csv('submission_6.csv', index=False)
```

Score
3.31287

Lgb

Lgb

- 可以reduce記憶體使用量
- 訓練速度br棒
- 可直接處理類別型資料

```
features = ['pickup_longitude', 'pickup_latitude', 'dropoff_longitude', 'dropoff_latitude',
            'passenger_count', 'abs_diff_longitude', 'abs_diff_latitude', 'H_Distance',
            'Year', 'Month', 'Date', 'Day of Week', 'Hour', 'pickup_place', 'dropoff_place'
]
target = ['fare_amount']
```

設定lgb model training要用的training dataset與evaluation dataset

GridSearchCV

https://www.kaggle.com/garethjns/microsoft-lightgbm-with-parameter-tuning-0-823
https://lightgbm.readthedocs.io/en/latest/Features.html#optimal-split-for-categorical-features

```
from sklearn.model_selection import train_test_split

train_X, test_X, train_y, test_y = train_test_split(X_features, X_target, test_size=0.2, random_state=0)
train_Xcv, test_Xcv, train_ycv, test_ycv = train_test_split(X_featurescv, X_targetcv, test_size=0.2, random_state=0)
```

```
import lightgbm as lgb
from sklearn.model_selection import GridSearchCV

#dataset
lgb_train = lgb.Dataset(train_X, train_y,
#                         categorical_feature=features,
                         silent=True,
                         free_raw_data=False)
lgb_eval = lgb.Dataset(test_X, test_y, reference=lgb_train)

#dataset
lgb_traincv = lgb.Dataset(train_Xcv, train_ycv,
#                         categorical_feature=features,
                         silent=True,
                         free_raw_data=False)
lgb_evalcv = lgb.Dataset(test_Xcv, test_ycv, reference=lgb_train)
```

用gridsearch進行調整參數

- 初始參數boosting_type, objective, metric

```
def lgb_gridmodel(train_X, test_X, train_y, test_y):
    import lightgbm as lgb
    from sklearn.model_selection import GridSearchCV
    #dataset
    lgb_train = lgb.Dataset(train_X, train_y)
    lgb_eval = lgb.Dataset(test_X, test_y, reference=lgb_train)
    #基本配備，初始參數boosting_type, objective, metric
    global params
    params = {
        'boosting_type': 'gbdt',
        'objective': 'regression',
        'metric': 'rmse',
    }
```

- 先調整num_leaves, max_depth

```
#cv
min_merror = float('Inf')
global best_params
best_params = {}
#num_leaves, max_depth
for num_leaves in range(20,200,5):
    for max_depth in range(3,8,1):
        params['num_leaves'] = num_leaves
        params['max_depth'] = max_depth
        cv_results = lgb.cv(
            params,
            lgb_train,
            seed=2019,
            stratified=False,
            nfold=5,
            metrics=['rmse'],
            early_stopping_rounds=10,
            verbose_eval=50
        )

        mean_merror = pd.Series(cv_results['rmse-mean']).min()
        boost_rounds = pd.Series(cv_results['rmse-mean']).argmin()

        if mean_merror < min_merror:
            min_merror = mean_merror
            best_params['num_leaves'] = num_leaves
            best_params['max_depth'] = max_depth
params['num_leaves'] = best_params['num_leaves']
params['max_depth'] = best_params['max_depth']
```

- 在處理max_bin, min_child_samples, min_child_weight

```
#max_bin, min_child_samples, min_child_weight(沒調)
for max_bin in range(5,255,5):
    for min_data_in_leaf in range(10,200,5):
        for min_child_weight in [0.001, 0.002, 0.003, 0.004, 0.005]:
            params['max_bin'] = max_bin
            params['min_data_in_leaf'] = min_data_in_leaf
            params['min_child_weight'] = min_child_weight
            cv_results = lgb.cv(
                params,
                lgb_train,
                seed=42,
                stratified=False,
                nfold=5,
                early_stopping_rounds=3,
                verbose_eval=50
            )
            mean_merror = pd.Series(cv_results['rmse-mean']).min()
            boost_rounds = pd.Series(cv_results['rmse-mean']).argmin()

            if mean_merror < min_merror:
                min_merror = mean_merror
                best_params['max_bin'] = max_bin
                best_params['min_data_in_leaf'] = min_data_in_leaf
                best_params['min_child_weight'] = min_child_weight
params['max_bin'] = best_params['max_bin']
params['min_data_in_leaf'] = best_params['min_data_in_leaf']
params['min_child_weight'] = best_params['min_child_weight']
```

- 决定feature_fraction, bagging_fraction, bagging_freq

```
#feature_fraction, bagging_fraction, bagging_freq
for feature_fraction in [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0]:
    for bagging_fraction in [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0]:
        for bagging_freq in range(0,50,5):
            params['feature_fraction'] = feature_fraction
            params['bagging_fraction'] = bagging_fraction
            params['bagging_freq'] = bagging_freq
            cv_results = lgb.cv(
                params,
                lgb_train,
                seed=42,
                stratified=False,
                nfold=5,
                metrics=['rmse'],
                early_stopping_rounds=3,
                verbose_eval=50
            )
            mean_merror = pd.Series(cv_results['rmse-mean']).min()
            boost_rounds = pd.Series(cv_results['rmse-mean']).argmin()

            if mean_merror < min_merror:
                min_merror = mean_merror
                best_params['feature_fraction'] = feature_fraction
                best_params['bagging_fraction'] = bagging_fraction
                best_params['bagging_freq'] = bagging_freq
params['feature_fraction'] = best_params['feature_fraction']
params['bagging_fraction'] = best_params['bagging_fraction']
params['bagging_freq'] = best_params['bagging_freq']
```

- 最後lambda_l1, lambda_l2, min_split_gain

```
#lambda_l1, lambda_l2, min_split_gain
for lambda_l1 in [0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0]:
    for lambda_l2 in [0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0]:
        for min_split_gain in [0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0]:
            params['lambda_l1'] = lambda_l1
            params['lambda_l2'] = lambda_l2
            params['min_split_gain'] = min_split_gain
            cv_results = lgb.cv(
                params,
                lgb_train,
                seed=42,
                stratified=False,
                nfold=5,
                metrics=['rmse'],
                early_stopping_rounds=3,
                verbose_eval=50
            )
            mean_merror = pd.Series(cv_results['rmse-mean']).min()
            boost_rounds = pd.Series(cv_results['rmse-mean']).argmin()

            if mean_merror < min_merror:
                min_merror = mean_merror
                best_params['lambda_l1'] = lambda_l1
                best_params['lambda_l2'] = lambda_l2
                best_params['min_split_gain'] = min_split_gain
params['lambda_l1'] = best_params['lambda_l1']
params['lambda_l2'] = best_params['lambda_l2']
params['min_split_gain'] = best_params['min_split_gain']
return best_params
```

採用top10 params進行training

```
params = {
        'boosting_type':'gbdt',
        'objective': 'regression',
        'nthread': -1,
        'verbose': 0,
        'num_leaves': 31,
        'learning_rate': 0.05,
        'max_depth': -1,
        'subsample': 0.8,
        'subsample_freq': 1,
        'colsample_bytree': 0.6,
        'reg_aplha': 1,
        'reg_lambda': 0.001,
        'metric': 'rmse',
        'min_split_gain': 0.5,
        'min_child_weight': 1,
        'min_child_samples': 10,
        'scale_pos_weight':1,
        'verbose':0
    }
```

```
lgb_train = lgb.Dataset(X_features, X_target, silent=True)
```

```
gbm = lgb.train(params, lgb_train, num_boost_round = 300)
print(gbm)
```

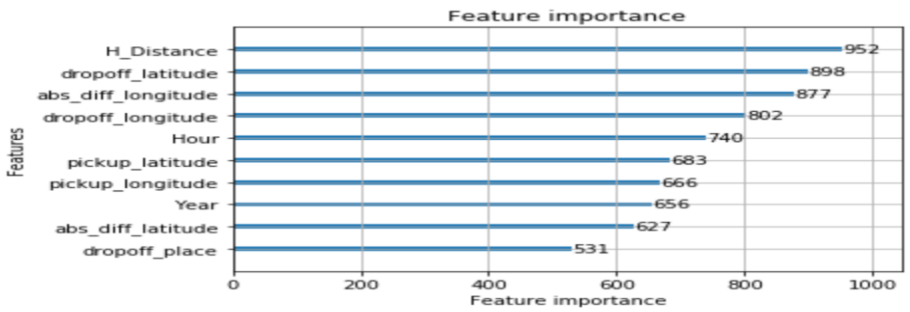
把lgb模型所訓練出來的資料，用gbm.predict進行預測

```
# predict
y_pred_lgb = gbm.predict(y_features, num_iteration = gbm.best_iteration)
print(y_pred_lgb)
```

```
[10.49015357 10.3011617   4.54784085 ... 54.76529484 20.619656
  6.98293117]
```

feature importance

```
#feature importance
ax = lgb.plot_importance(gbm, max_num_features=10)
```



結果示意

結果(未轉換參數)

Score

3.81496

結果(未轉換參數)

```
params['learning_rate']=0.05
params
```

```
{'boosting_type': 'gbdt',
 'objective': 'regression',
 'metric': 'rmse',
 'num_leaves': 40,
 'max_depth': 7,
 'max_bin': 5,
 'min_data_in_leaf': 55,
 'min_child_weight': 0.001,
 'feature_fraction': 0.9,
 'bagging_fraction': 0.1,
 'bagging_freq': 0,
 'lambda_l1': 0.9,
 'lambda_l2': 0.4,
 'min_split_gain': 0.6,
 'learning_rate': 0.05}
```

結果(轉換參數)

Score

3.20180

CatBoostRegressor

CatBoostRegressor

- 可直接處理類別型資料
- 有較好的方式防止overfitting(IncToDec)
- training過程有優美的圖供人觀賞

```
features = ['pickup_longitude', 'pickup_latitude', 'dropoff_longitude', 'dropoff_latitude',
            'passenger_count', 'month', 'week', 'time_interval', 'weekend', 'holiday',
            'pickup_place', 'dropoff_place', 'area_crossing',
            'linear_distance']
target = ['fare_amount']
```

特別標示categorical_features

```
categorical_features_indices = np.where(X_features.dtypes != np.float)[0]
```

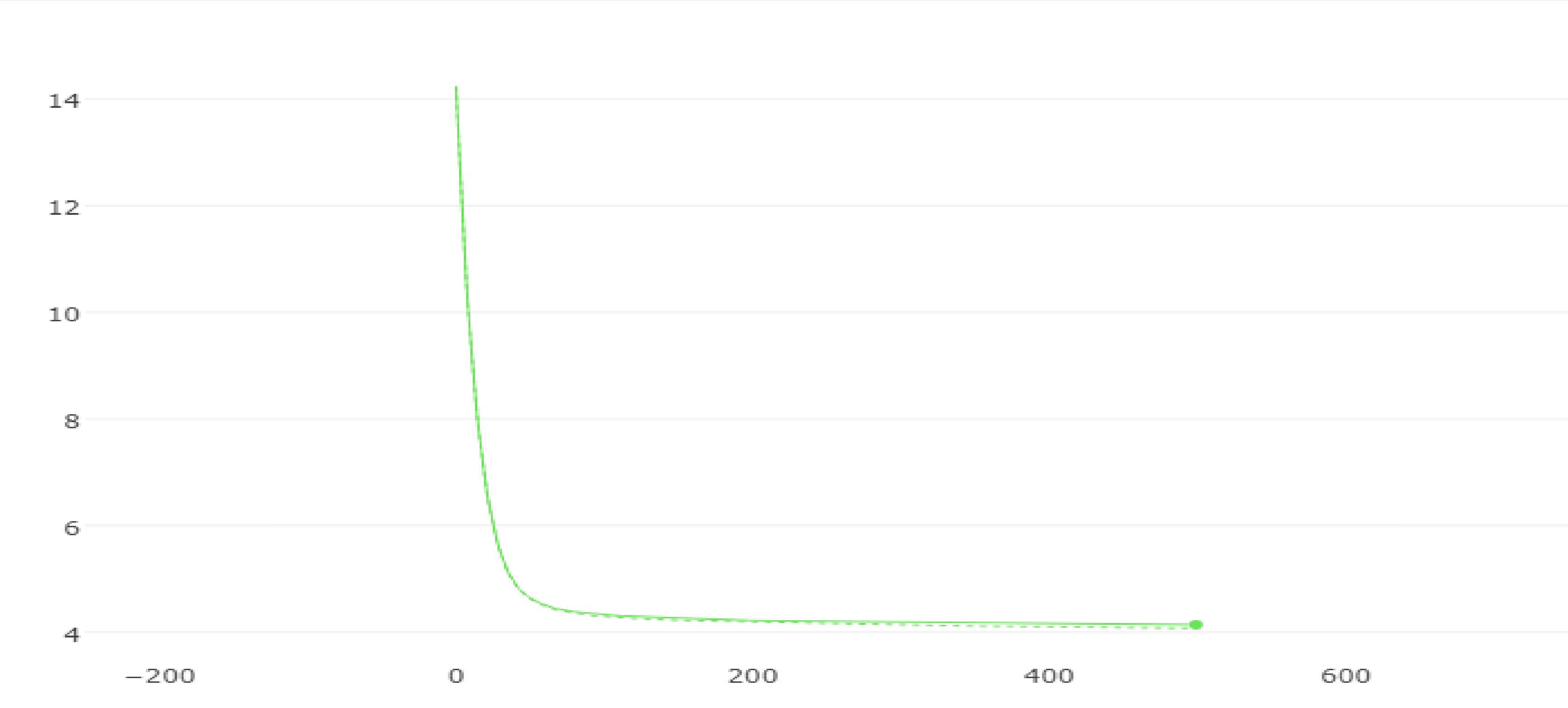
使用CatBoostRegressor

```
train_pool  = Pool(X_train, y_train, cat_features=categorical_features_indices)
validate_pool = Pool(X_validation, y_validation, cat_features=categorical_features_indices)
```

```
#default parameters
model = CatBoostRegressor(
        learning_rate=0.05, #default 0.03
        depth=10, #default 6
        loss_function='RMSE',
        eval_metric='RMSE',
        random_seed=0,
        logging_level='Verbose',
        use_best_model=True,
        od_type='IncToDec',
        od_pval=0.001,
        od_wait=40
)
```

```
%%time
model.fit(
        X_train, y_train,
        cat_features=categorical_features_indices,
        eval_set=validate_pool,
        logging_level='Verbose',
#        plot=True #可看圖!!!
);
```

視覺化training流程



predict結果

```
y_pred_cb = model.predict(X_test)
```

feature importance

```
feature_importances = model.get_feature_importance(train_pool)

feature_names = X_train.columns
for score, name in sorted(zip(feature_importances, feature_names), reverse=True):
    print('{}: {}'.format(name, score))
```

```
linear_distance: 46.043432838422646
dropoff_longitude: 15.983201338876535
dropoff_latitude: 9.016162293332544
pickup_longitude: 7.920119092640227
dropoff_place: 5.441035758746613
pickup_latitude: 4.215268656953611
time_interval: 3.5611228058684246
pickup_place: 2.7105976769639937
month: 1.8984989336625384
week: 1.497503565559565
passenger_count: 0.660073886317301
weekend: 0.49864350010183844
area_crossing: 0.4636963036820582
holiday: 0.09064334887209347
```

結果

調參前

Score
3.91763

調參後

Score
3.58458

三項模型的結果比對

RandomForestRegressor

Score
3.31287

Lgb

Score
3.20180

CatBoostRegressor

Score
3.58458

Lgb + RandomForestRegressor

Score
3.12304

組員分工

- **03152138 張永霖**
- **04155136 馮顯典**
- **04170104 游惠如**
- **04170116 沈芳儀**
- **04170118 洪聆紜**

THANK YOU

Please click here to modify the text for example
The text here you may post texts

感谢一路有你