Machine Learning Engineer Nanodegree

Capstone Project

Benjamin DK Luong

November 11, 2018

I. Definition

Project Overview

Expedia wants to take the proverbial rabbit hole out of hotel search by providing personalized hotel recommendations to their users. This is no small task for a site with hundreds of millions of visitors every month!

Currently, Expedia uses search parameters to adjust their hotel recommendations, but there aren't enough customer specific data to personalize them for each user. In this competition, Expedia is challenging Kagglers to contextualize customer data and predict the likelihood a user will stay at 100 different hotel groups.

Problem Statement

Planning your dream vacation, or even a weekend escape, can be an overwhelming affair. With hundreds, even thousands, of hotels to choose from at every destination, it's difficult to know which will suit your personal preferences. Should you go with an old standby with those pillow mints you like, or risk a new hotel with a trendy pool bar?

Metrics

In order to compare between models, I use accuracy score and F-score to find out the best model.

According to Exsilio Solutions:

	Predicted class		
Actual Class		Class = Yes	Class = No
	Class = Yes	True Positive	False Negative
	Class = No	False Positive	True Negative

True Positive = TP
True Negative = TN

False Positive = FP
False Negative = FN

Accuracy - Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. One may think that, if we have high accuracy then our model is best. Yes, accuracy is a great measure but only when you have symmetric datasets where values of false positive and false negatives are almost same. Therefore, you have to look at other parameters to evaluate the performance of your model.

$$\text{Accuracy} = TP+TN/TP+FP+FN+TN$$

F1 score - F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall.

$$\text{F1 Score} = 2*(\text{Recall} * \text{Precision}) / (\text{Recall} + \text{Precision})$$

II. Analysis

(approx. 2-4 pages)

Data Exploration

- The datatypes are not correct for variables: date_time, srch_ci, and srch_co. They are should be in datetime, but they are string. I have to change them to datetime datatype.
- After I changed them to datetime datatype, I created a new variable called vac_length which is the difference between srch_ci and srch_co. This variable shows the length of each vacation. For example, a vacation length is 4 days, 5 days, etc.
- I want to know what month people want to have their vacation in. I believe when they go on vacation will play an importance role where they will stay. The place people stay in summer would be different the place they stay in winter.
- I drop some variables that are not necessary such as: date_time, user_id, srch_ci,srch_co, srch_destination_id , srch_destination_type_id, and orig_destination_distance. All id variables will not contain any information

that show customers' preferences. Orig_destination_distance variable has a lot of NAN or Null, so I drop it.

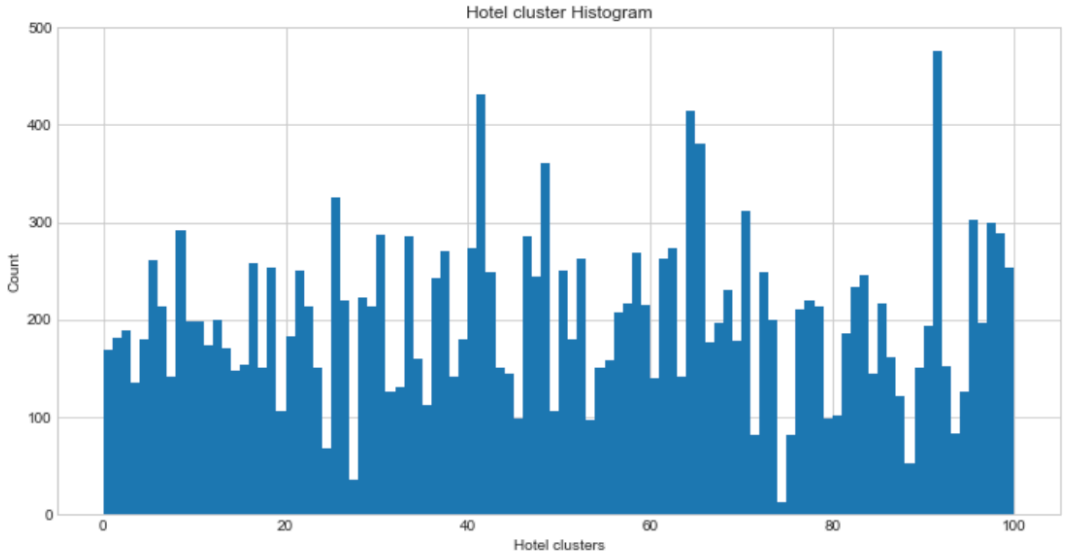
- After cleaning up the dataset, I check correlations between all variables and hotel_cluster variable. I see there is no correlation between them.

```
user_location_country   -0.030248
site_name               -0.019771
hotel_country           -0.014768
user_location_city      -0.005499
vac_month               -0.005296
channel                 -0.004559
is_mobile                0.000157
posa_continent           0.001789
srch_rm_cnt              0.001941
hotel_continent          0.002260
srch_adults_cnt          0.013241
srch_children_cnt        0.013741
hotel_market             0.022189
user_location_region     0.023721
vac_length               0.053685
is_package               0.066405
hotel_cluster            1.000000
Name: hotel_cluster, dtype: float64
```

- I drop all NaN or null on the whole dataset, so I can start implement models on it.

Exploratory Visualization

- The plot below shows hotel_cluster distribution. It's totally random.



Algorithms and Techniques

Because of finding hotel_cluster, I will use classification supervised learning algorithms:

In this section, you will need to discuss the algorithms and techniques you intend to use for solving the problem. You should justify the use of each one based on the characteristics of the problem and the problem domain. Questions to ask yourself when writing this section:

- clf_A = SVC(random_state = 50)
- clf_B = RandomForestClassifier(random_state = 50)
- clf_C = SGDClassifier(random_state = 50)
- clf_D = KNeighborsClassifier(n_neighbors=5)
- clf_E = LogisticRegression(random_state = 50)
- clf_F = GaussianNB()
- clf_G = XGBClassifier(random_state = 50)

I will choose the best algorithm to solve the problem.

When I look at the dataset from Kaggle, it has train set and test set. However, the test set doesn't have hotel_cluster variable because Kaggle will tell our models with the test set. In order to train and test our model, I split the train set into 2 parts: one is for training, another one is for testing and comparing our

models. Moreover, the train set is too large, so I just use a part of it to train and test our model to save time.

```
# Import train_test_split
from sklearn.cross_validation import train_test_split

# Split the 'features' and 'income' data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features,
                                                    cluster,
                                                    test_size = 0.2,
                                                    random_state = 0)

# Show the results of the split
print("Training set has {} samples.".format(X_train.shape[0]))
print("Testing set has {} samples.".format(X_test.shape[0]))
```

I fit the X_train and y_train into all algorithms above, and test the algorithms with X_test to get prediction values. Finally I will compare the prediction values against y_test using accuracy score and F score.

- accuracy_score(y_test,predictions_test)
- fbeta_score(y_test,predictions_test,beta = 0.5,average = 'micro')

Benchmark

Here is the results after training and testing our models.

```
'GaussianNB':
{'acc_test': 0.064032016008004,
  'acc_train': 0.07666666666666666,
  'f_test': 0.064032016008004,
  'f_train': 0.07666666666666666,
  'pred_time': 0.047205209732055664,
  'train_time': 0.01756906509399414},

'KNeighborsClassifier':
{'acc_test': 0.224112056028014,
  'acc_train': 0.48,
  'f_test': 0.224112056028014,
  'f_train': 0.48,
  'pred_time': 0.05237698554992676,
  'train_time': 0.02214813232421875},

'LogisticRegression':
{'acc_test': 0.06678339169584792,
  'acc_train': 0.09333333333333334,
  'f_test': 0.06678339169584792,
  'f_train': 0.09333333333333334,
  'pred_time': 0.005501270294189453,
  'train_time': 28.152716159820557},
```

```
'RandomForestClassifier':
{'acc_test': 0.2933966983491746,
  'acc_train': 0.67,
  'f_test': 0.2933966983491746,
  'f_train': 0.6700000000000002,
  'pred_time': 0.0604400634765625,
  'train_time': 0.32732200622558594},

'SGDClassifier':
{'acc_test': 0.007503751875937969,
  'acc_train': 0.0033333333333333335,
  'f_test': 0.007503751875937968,
  'f_train': 0.0033333333333333335,
  'pred_time': 0.008719921112060547,
  'train_time': 0.5117590427398682},

'SVC':
{'acc_test': 0.2978989494747374,
  'acc_train': 0.6266666666666667,
  'f_test': 0.2978989494747374,
  'f_train': 0.6266666666666667,
  'pred_time': 8.815993785858154,
  'train_time': 17.040926933288574},

'XGBClassifier':
{'acc_test': 0.21060530265132565,
  'acc_train': 0.37333333333333335,
  'f_test': 0.21060530265132565,
  'f_train': 0.37333333333333335,
  'pred_time': 4.808354139328003,
  'train_time': 107.8650450706482}}
```

The results show that RandomForestClassifier algorithm give highest score with shortest time compare to others. I will choose RandomForestClassifier as my model. I will use GridsearchCV to improve the performance of the model.

III. Methodology

Data Preprocessing

I cleaned up the dataset in Data Exploration section, so I can analyze the dataset accurately. Moreover, I can feed cleaned data into my model. Please look at Data Exploration section to see how I cleaned up the data.

Implementation

Because I am using many algorithms, so I define a function to automate my workflow. Here is the function:

```
def train_predict(learner, sample_size, X_train, y_train, X_test, y_test):

    results = {}

    start = time() # Get start time

    learner = learner.fit(X_train[:sample_size], y_train[:sample_size])

    end = time()

    results['train_time'] = end - start

    start = time() # Get start time

    predictions_test = learner.predict(X_test)

    predictions_train = learner.predict(X_train[:300])

    end = time() # Get end time

    results['pred_time'] = end - start

    results['acc_train'] = accuracy_score(y_train[:300],predictions_train)

    results['acc_test'] = accuracy_score(y_test,predictions_test)

     results['f_train'] = fbeta_score(y_train[:300],predictions_train,beta = 0.5,average =
'micro')

    results['f_test'] = fbeta_score(y_test,predictions_test,beta = 0.5,average = 'micro')

     print("{} trained on {} samples.".format(learner.__class__.__name__, sample_size))

     return results
```

I just need to feed all algorithms into the train_predict function to get results: accuracy score, F score, and training time. Here how I do it:

```
results = {}

for clf in [clf_A, clf_B, clf_C, clf_D, clf_E,clf_F,clf_G]:

    clf_name = clf.__class__.__name__

    results[clf_name] = train_predict(clf, len(y_train), X_train, y_train, X_test, y_test)
```

The results are showed in the Benchmark section. Please take a look.

Refinement

In order to improve our chosen model (RandomForestClassifier), I use GridsearchCV to find the best model. The GridsearchCV will improve our model by changing the parameters:

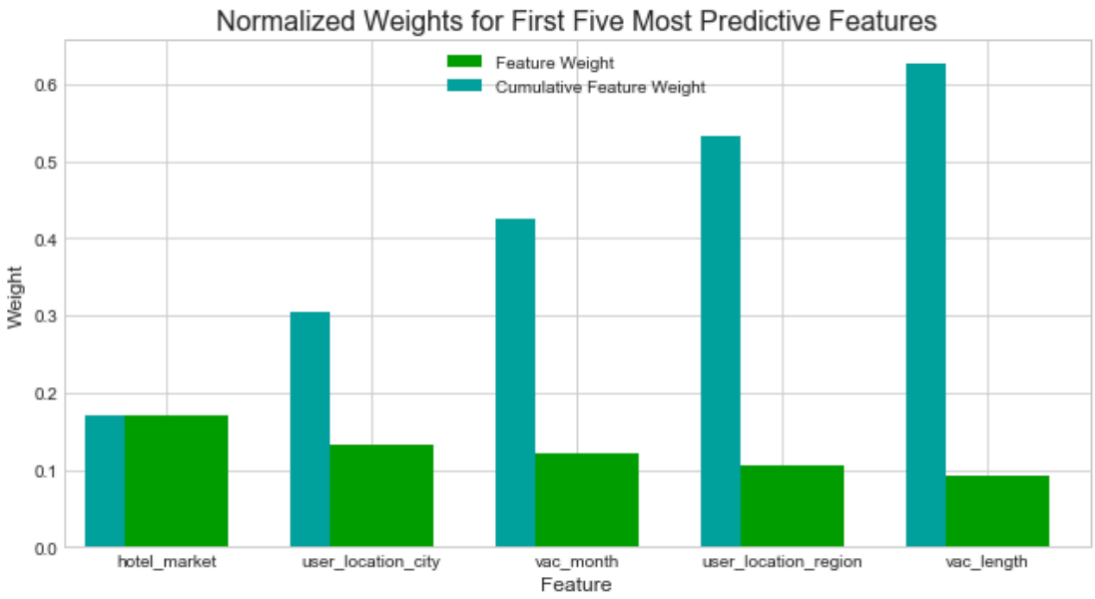
```
parameters = {'n_estimators': [3,50,100],
              'max_depth': [5,10],
              'min_samples_split':[2,5,7,10],
              'max_features':["auto","log2"]}
```

I fit the data into the best model again, and use it to make prediction.

```
grid_obj = GridSearchCV(clf, parameters, scoring = scorer)
grid_fit = grid_obj.fit(X_train,y_train)
best_clf = grid_fit.best_estimator_
predictions = (clf.fit(X_train, y_train)).predict(X_test)
best_predictions = best_clf.predict(X_test)
```

Now I have the best model. However, I have been using all variables to feed into the model. I can reduced the data by using feature importance, so the data will be smaller, train faster, but it still has same performance.

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(random_state = 50)
model.fit(X_train, y_train)
importances = model.feature_importances_
```



Above is the top 5 variables that I can use in the dataset.

The code below shows that I use only hotel_market, user_location_city, vac_month, user_location_region, and vac_length in the dataset to feed into the best model to get good performance.

```
from sklearn.base import clone

X_train_reduced = X_train[X_train.columns.values[(np.argsort(importances)[::-1])[:5]]]

X_test_reduced = X_test[X_test.columns.values[(np.argsort(importances)[::-1])[:5]]]

clf = (clone(best_clf)).fit(X_train_reduced, y_train)

reduced_predictions = clf.predict(X_test_reduced)

print("Final Model trained on full data\n------")

print("Accuracy on testing data: {:.4f}".format(accuracy_score(y_test, best_predictions)))

print("F-score on testing data: {:.4f}".format(fbeta_score(y_test, best_predictions, beta = 0.5,average = 'micro')))

print("\nFinal Model trained on reduced data\n------")

print("Accuracy on testing data: {:.4f}".format(accuracy_score(y_test, reduced_predictions)))
```

```
print("F-score on testing data: {:.4f}".format(fbeta_score(y_test,
reduced_predictions, beta = 0.5,average = 'micro')))
```

Here is the result:

```
Final Model trained on full data
------
Accuracy on testing data: 0.2401
F-score on testing data: 0.2401

Final Model trained on reduced data
------
Accuracy on testing data: 0.2331
F-score on testing data: 0.2331
```

I can use the best model with reduced dataset to get accuracy score and F-score that close to the full data ‘s performance.

IV. Results

Model Evaluation and Validation

	acc_test	acc_train	f_test	f_train	pred_time	train_time
GaussianNB	0.0640320160080004	0.0766	0.064	0.0766	0.0472	0.017
KNeighborsClassifier	0.224	0.48	0.224	0.48	0.05	0.022
LogisticRegression	0.066	0.093	0.0667	0.093	0.0055	28.15
RandomForestClassifier	**0.2933**	**0.67**	**0.2933**	**0.067**	**0.0604**	**0.327**
SGDClassifier	0.007	0.003	0.007	0.003	0.008	0.511

SVC	0.297	0.626	0.297	0.626	8.8	17.04
XGBClassifier	0.2106	0.373	0.21	0.373	4.8	107.86

We can see that RandomForestClassifier is the best compare to other algorithms. Here is the best model information:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion=
'gini',
            max_depth=10, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=1
,
            oob_score=False, random_state=50, verbose=0, warm_start=
False)
```

The model has tested with unseen data because I split data into train set and test set.

Justification

After comparing between many algorithms, I chose RandomForestClassifier. However, the accuracy score and F-score are so low (0.2933) . It does a lot better on training set (0.67). It means it corrects 67% with it is fed with train data. If it is fed with unseen data from testing set, the accuracy score failed to 29.33%. It means it correct only 29.33% . We need to find a solution to improve the model's performance even though we tried GridSearchCV and feature importance.

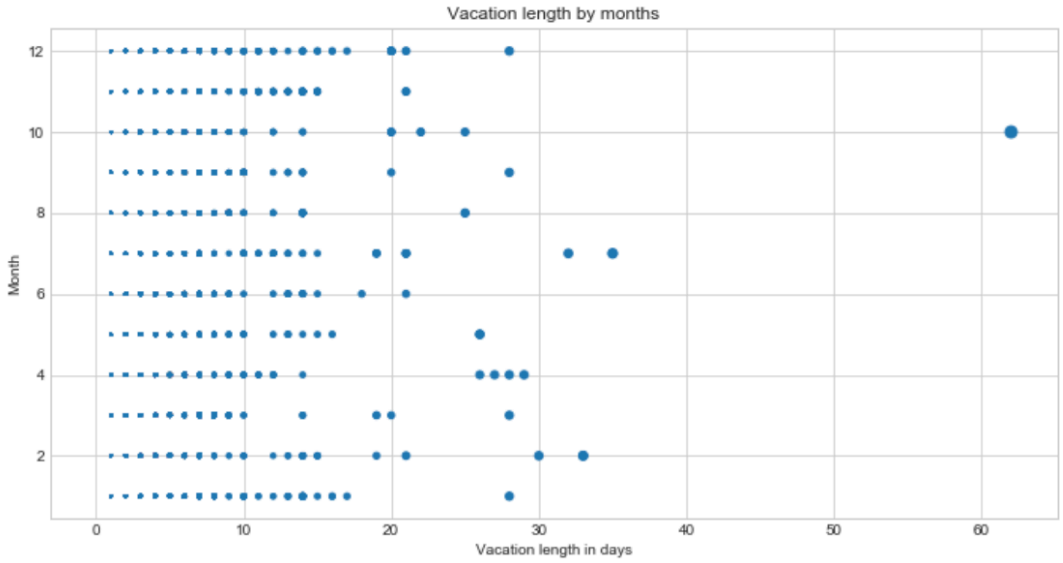
I don't think this model is strong enough to solve the problem or compete with other algorithms on Kaggle. The best score on Kaggle is 0.60443. We have a lot of room for improvements.

V. Conclusion

Free-Form Visualization

There is no correlations between all variables. It's hard to plot them and gain insight from them. For example:

I found this graph interesting when I plot vacation length against vacation month to see if people stay any longer in a given month. The graph below shows that the vacation length are pretty them same across every month.



Reflection

This project is hard because there is no correlations between all variables. After using algorithms to cluster the data points, I still have low accuracy score. I created 2 new variables: vac_length and vac_month. Both of them are in the top 5 of importance features, but they can not improve the score.

Improvement

I believe there are ways to improve the accuracy score, but I don't know yet. I tried feature importance and gridsearchcv to get the best algorithm. Unfortunately, the score is still low. It corrects about 30% of the time.

I could train my model on full data to see if it's improved or not. The dataset is too large (37,670,293 rows). I used only 20,000 rows in the dataset. However, I don't think it's the main problem here.

The goal is I have to find a new way to improve my best model or I have to use different algorithms.