Google Analytics Customer Revenue Prediction

2018 Fall Data Science Capstone DATS 6501

Cheng Miao

Professor: Amir Hossein Jafari

Columbian College of Arts & Sciences

Introduction and Object

- This project is created to predict the Google Store Customer Revenue based on machine learning methods(Predict how much GStore customers will spend).
- The outcome will be more actionable operational changes and a better use of marketing budgets for those companies who choose to use data analysis on top of GA data.

- The object for this project is to implement learned machine learning knowledge in class and explore new knowledge.
- Compare the advantages and disadvantages of using the model in the project.
- Make data analysis and give advice to some companies.

Dataset

- Google Analytics Customer Revenue Prediction
- Dataset link: https://www.kaggle.com/c/ga-customer-revenue-prediction/data
- The data has 50 columns, with “transaction Revenue” as the target to explore the relationship between the rest of the dataset.
- Dataset has factors such as Visitor ID, Country, Mobile, Week, etc.

Problem Statement

- Which factors will have higher impact on model?
- How does model performance?
- Any advice to companies?

Packages:Numpy, Pandas, Matplotlib, Sklearn, Json, Seaborn, torch, lightGBM, etc

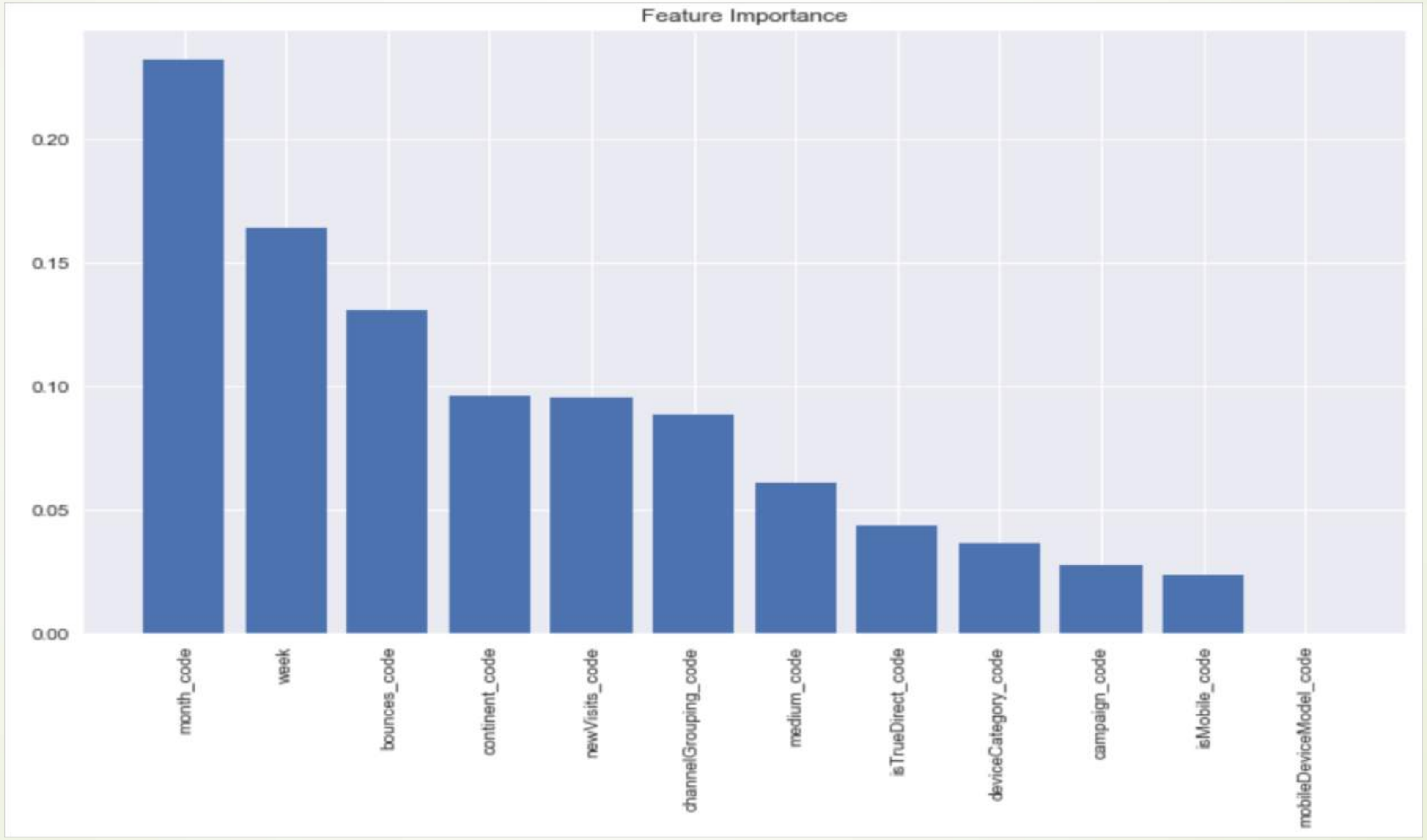
Dataset

- How dataset looks like

```
1  data.head()
```

	channelGrouping	date	fullVisitorId	sessionId	visitId	visitNumber	visitStartTime	browser	deviceCategory	isMobile
0	Organic Search	2016-09-02	1.131660e+18	1131660440785968503_1472830385	1472830385	1	1472830385	Chrome	desktop	False
1	Organic Search	2016-09-02	3.773060e+17	377306020877927890_1472880147	1472880147	1	1472880147	Firefox	desktop	False
2	Organic Search	2016-09-02	3.895550e+18	3895546263509774583_1472865386	1472865386	1	1472865386	Chrome	desktop	False
3	Organic Search	2016-09-02	4.763450e+18	4763447161404445595_1472881213	1472881213	1	1472881213	UC Browser	desktop	False
4	Organic Search	2016-09-02	2.729440e+16	27294437909732085_1472822600	1472822600	2	1472822600	Chrome	mobile	True

Data Preprocessing: Random Forest-Feature Importance



Data Preprocessing

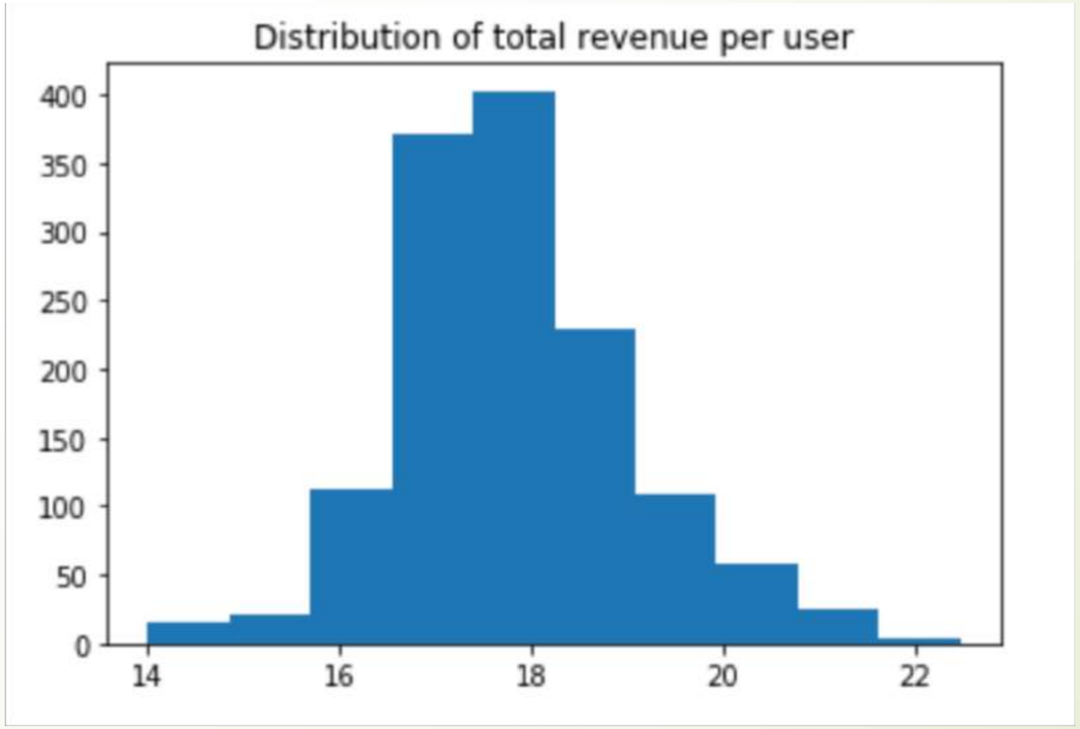
- Data Preprocessing really important, it may takes around 60% time of a project.

```
for r in data.columns:
    a = data[r].value_counts()
    if len(a)<2:
        print(r)
        print(a)
        print('----')

# value is constant, get rid of these, I have 38 columns left.
data = data.drop(['socialEngagementType','browserSize','browserVersion','flashVersion',\
                  'language','mobileDeviceBranding','mobileDeviceInfo','mobileDeviceMarketingName',\
                  'mobileInputSelector','operatingSystemVersion','screenResolution','screenColors','cityId',\
                  'latitude','longitude','networkLocation','visits','adwordsClickInfo.criteriaParameters','ca
```

```
for r in data.columns:
    a = len(data[r][pd.isnull(data[r])])/len(data)
    if a>0.8:
        print(r)
        print(a)
        print('----')

# starting drop columns
data = data.drop(['adContent','adwordsClickInfo.adNetworkType','adwordsClickInfo.gclId','adwordsClickInfo.isVide
```

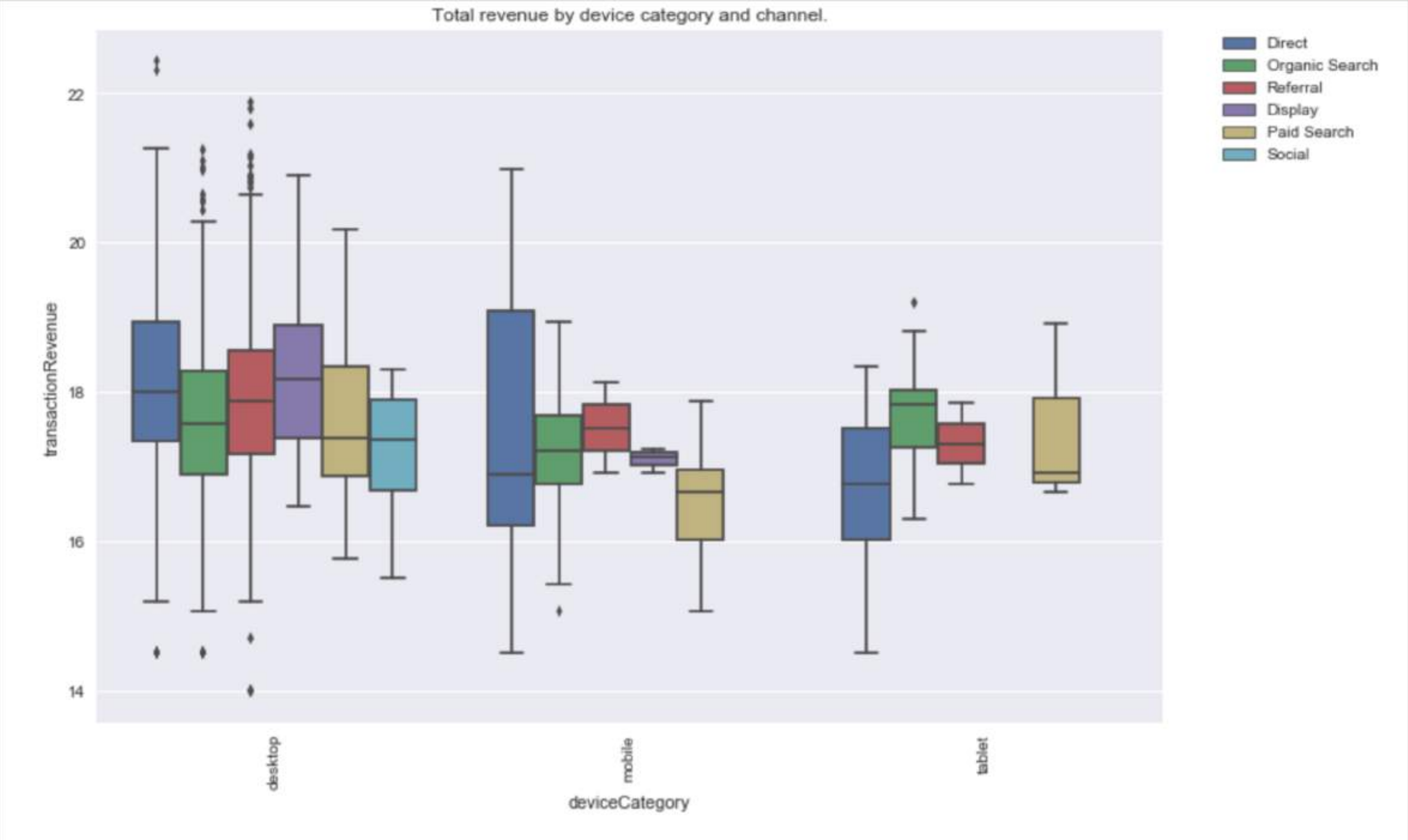
Exploratory Data Analysis (EDA)

- it is more useful to see a distribution of total revenue per user
- See the total revenue per user focus on 16 to 19 presenting a normal distribution.



Exploratory Data Analysis (EDA)

- Revenue comes mostly from desktops. Social, Affiliates and others aren't as profitable as other channels

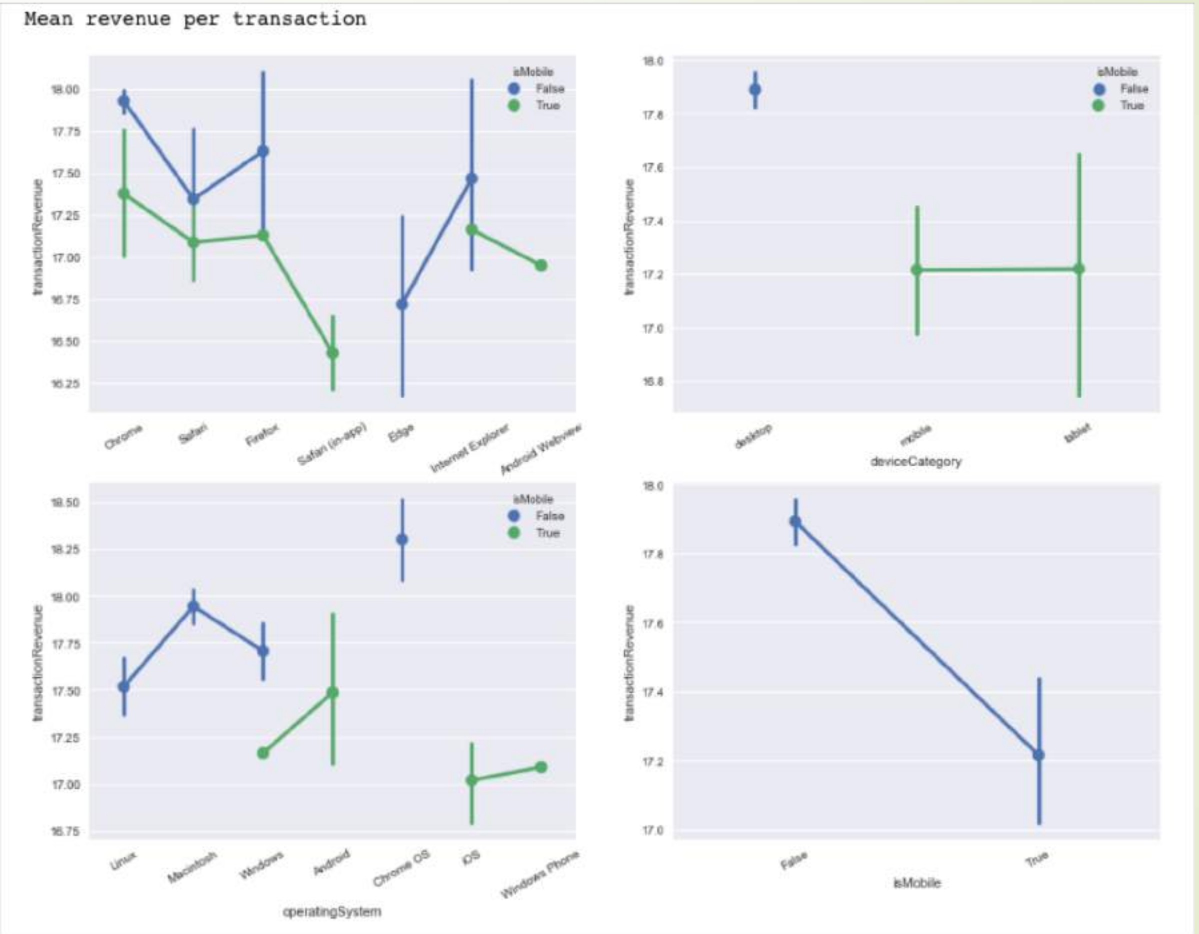


Exploratory Data Analysis (EDA)

- It is that trends of non-paying users and paying users of paid transactions are almost similar.
- There are several periods when the number of non-paying users was significantly higher that the number of paying users.

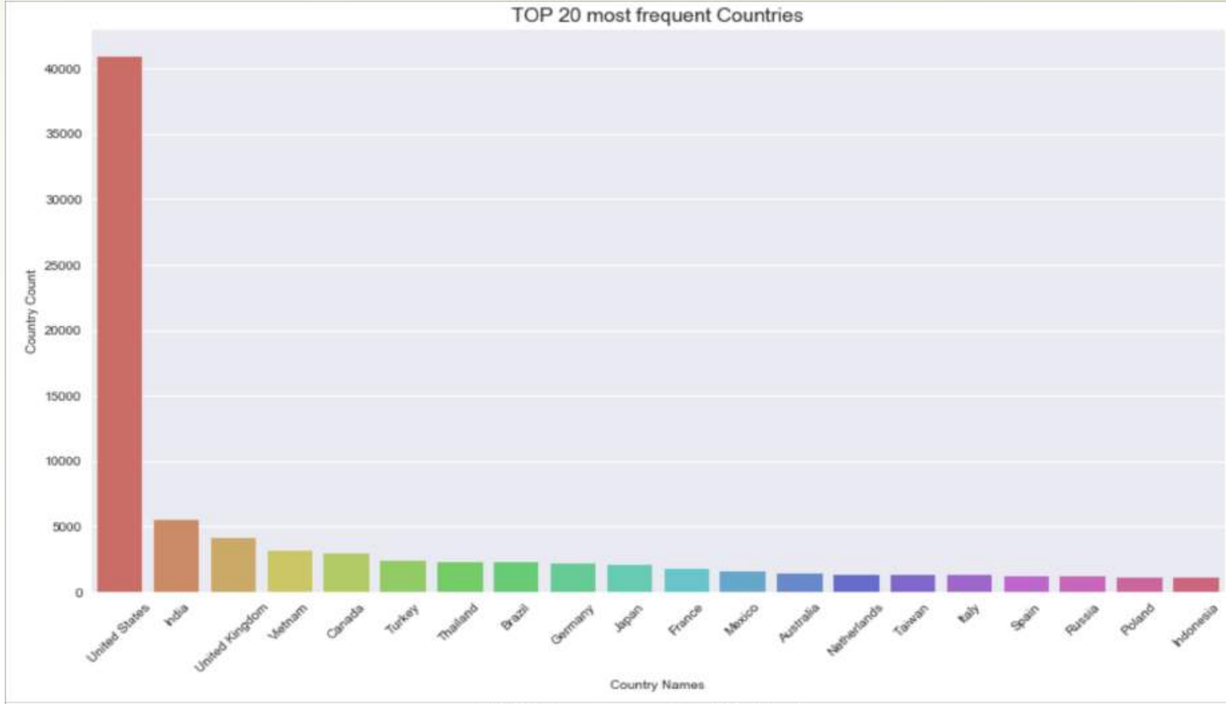
Exploratory Data Analysis (EDA)

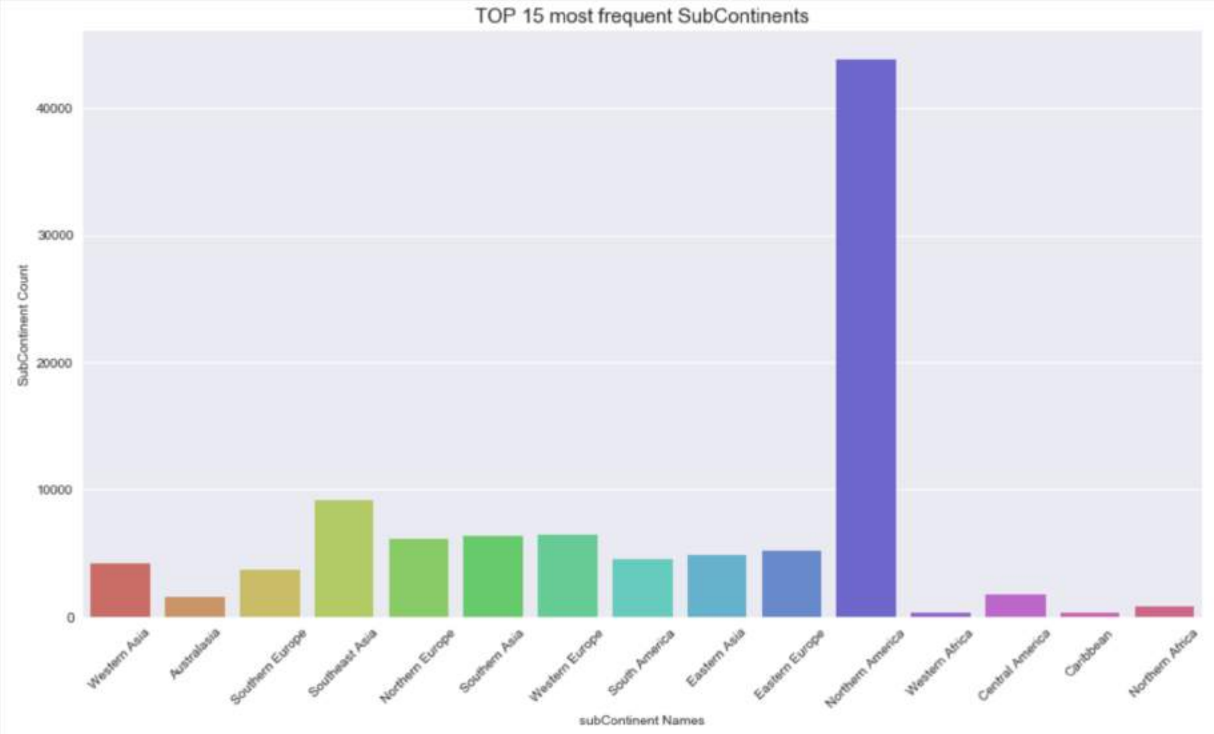
- The top-left graph shows devices on Chrome, Safari and Firefox could bring profit.
- Left corner graph looks that devices on Chrome OS and Macs bring most profit.
- Top-right graph, mobile and tablet bring most profit.



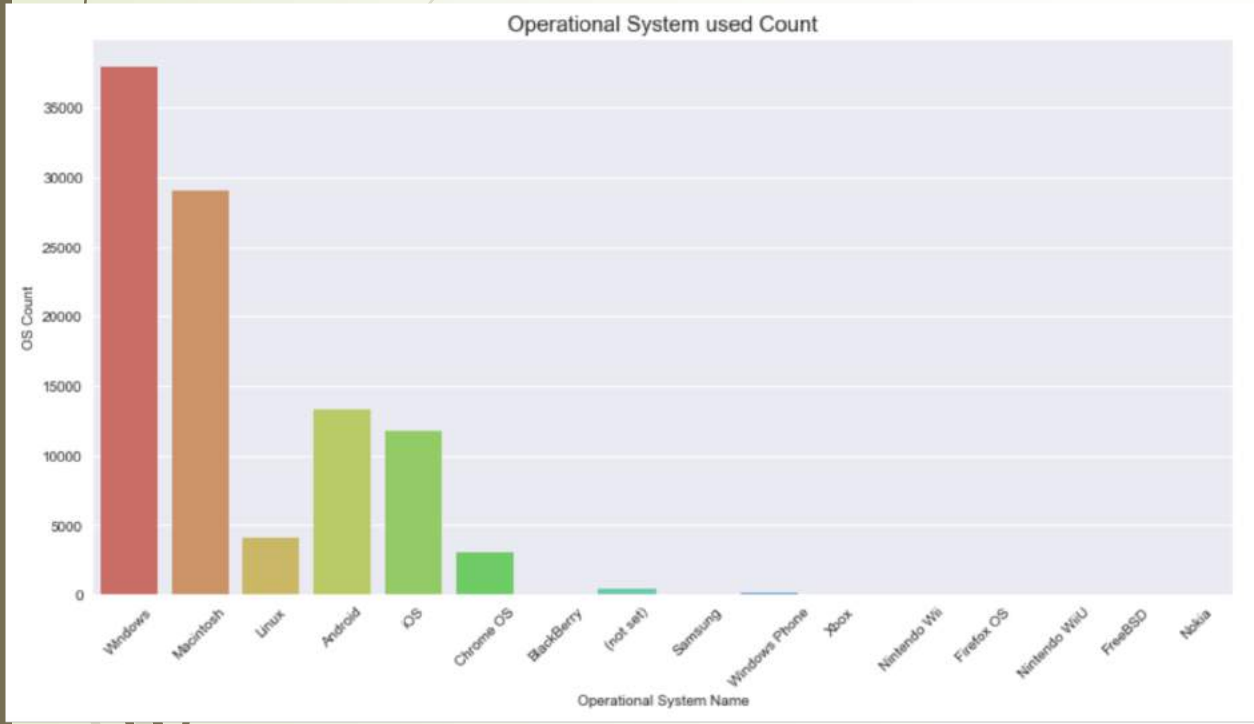
Exploratory Data Analysis (EDA)

- The above graph shows the top 20 most frequent countries. The United States stands out in the world and has the highest frequency.
- This bottom graph shows the top 15 most frequent Sub Continents, the Northern America has the most frequencies in the world.





Exploratory Data Analysis (EDA)



- This graph shows the operational system used count bar chart. Windows and Mac operational system there are the top two operational system

Methodology

- Light GBM
- Random Forest
- Multi-layer perceptron - Feed forward Neural Network

Light GBM

- Used grid search to select the appropriate model parameters.
- Learning rate is 0.1 and the n_estimators selection is 20.
- Bring this parameter into the LGBM Regressor to train my model.

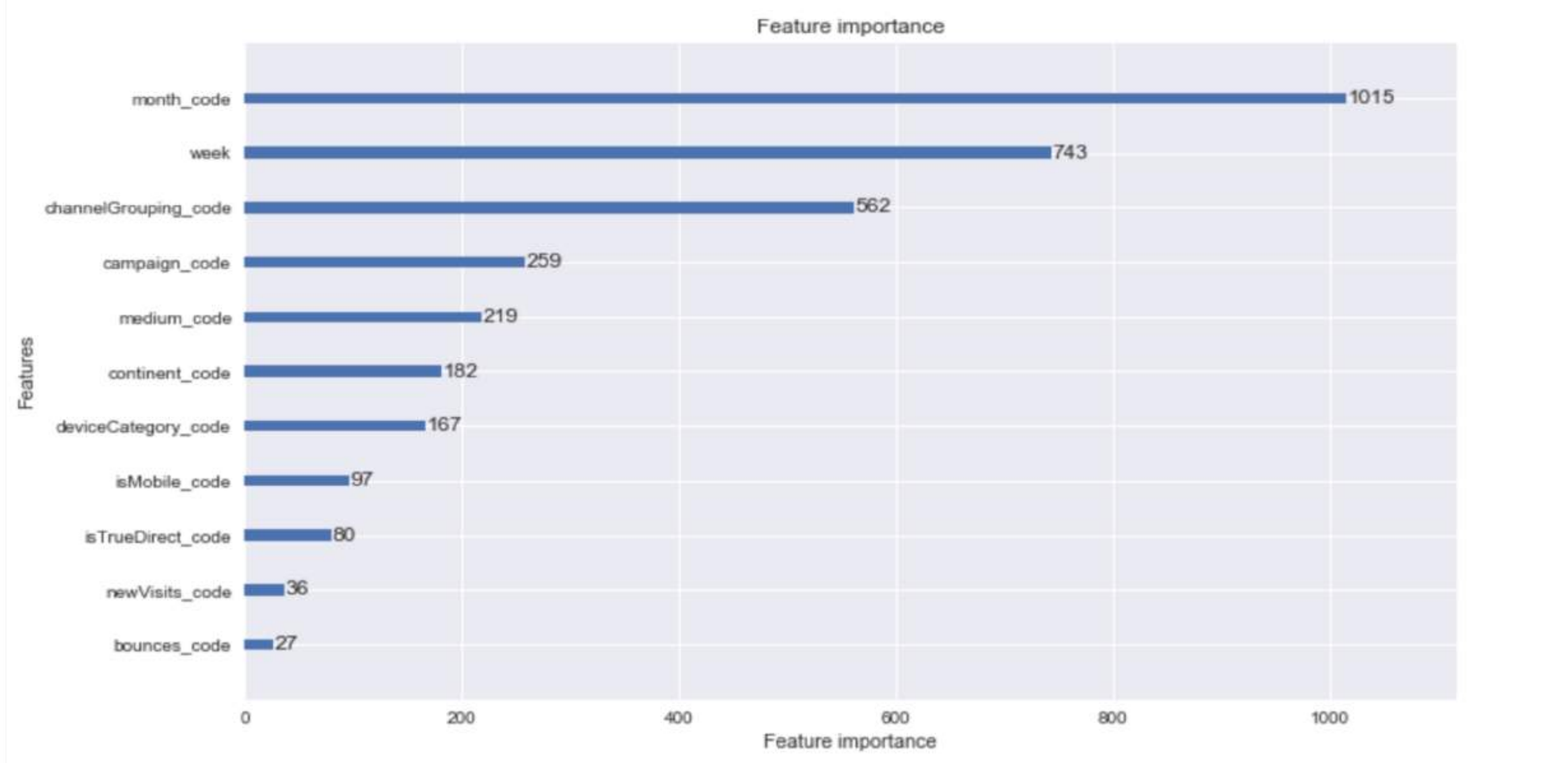
```
estimator = lgb.LGBMRegressor(num_leaves=31)

param_grid = {
    'learning_rate': [0.01, 0.05, 0.1],
    'n_estimators': [10, 20, 40],
}

gbm = GridSearchCV(estimator, param_grid, cv=3)
gbm.fit(X_train, y_train)

print('Best parameters found by grid search are:', gbm.best_params_)
```

```
Starting predicting...
The rmse of prediction is: 2.039396145191303
Feature importances: [127, 76, 25, 12, 0, 44, 36, 18, 14, 38, 14, 136]
Starting training with custom eval function...
[1]     valid_0's l2: 4.38725   valid_0's RMSLE: 0.382336
Training until validation scores don't improve for 5 rounds.
[2]     valid_0's l2: 4.34299   valid_0's RMSLE: 0.378514
[3]     valid_0's l2: 4.30754   valid_0's RMSLE: 0.377066
[4]     valid_0's l2: 4.27729   valid_0's RMSLE: 0.37687
[5]     valid_0's l2: 4.2536    valid_0's RMSLE: 0.377813
[6]     valid_0's l2: 4.23376   valid_0's RMSLE: 0.379385
[7]     valid_0's l2: 4.21987   valid_0's RMSLE: 0.381486
[8]     valid_0's l2: 4.20675   valid_0's RMSLE: 0.383569
[9]     valid_0's l2: 4.19588   valid_0's RMSLE: 0.385673
Early stopping, best iteration is:
[4]     valid_0's l2: 4.27729   valid_0's RMSLE: 0.37687
Starting predicting...
The rmsle of prediction is: 0.3768701275205428
Best parameters found by grid search are: {'learning_rate': 0.1, 'n_estimators': 20}
```

```
gbm=lgb.LGBMRegressor(num_leaves=31,
                      learning_rate=0.1,
                      n_estimators=20)

gbm.fit(X_train, y_train,
        eval_set=[(X_test, y_test)],
        eval_metric='l1',
        early_stopping_rounds=5)

print('Starting predicting...')
# predict
y_pred = gbm.predict(X_test, num_iteration=gbm.best_iteration_)
```

Light GBM – Feature Importance

```
lgb.plot_importance(model, figsize=(12, 7))
plt.show()
```



Random Forest

```python
from sklearn.ensemble import RandomForestRegressor
from sklearn import metrics

rf=RandomForestRegressor()
rf.fit(X_train, y_train)
rf_pred_test=rf.predict(X_test)
print('prediction:',rf_pred_test)
print('RMSE',np.sqrt(metrics.mean_squared_error(y_test,rf_pred_test)))
```

```
prediction: [0.97870904 0.         1.71891184 ... 0.         0.         0.        ]
RMSE 2.097921376837861
```

Principal Component Analysis (PCA)

- Use PCA to make the data on dimensionality reduction

```
[-1.21675246  0.32847542  1.82301643 -1.77005558  0.85666901  0.72287247
 -0.52267115  1.12312211]
```

Multilayer Feed-Forward Neural Network

- Convert the data into a torch format suitable for the Pytorch framework, in order to work on training and make prediction.
- Before inputting to the neural network as input, I have to convert the data to float

```
#Convert data for pytorch training more easy
X_train=torch.from_numpy(np.array(X_train))
y_train=(torch.from_numpy(np.array(y_train))).view(59999,1)
X, y = Variable(X_train,requires_grad=True), Variable(y_train,requires_grad=True)
```

```
# convert data type float
x=X.float()
y=y.float()
```

Multilayer Feed-Forward Neural Network: Define Model

- Define the Neural Network model refers to a lot of research online
- n_feature s=8, 9 is hidden layer.
- This parameter can be conditional.
- The hidden layer can be adjusted from 1 to 100. Due to its limited capacity, it is temporarily set to 9. The output is 1

```python
import torch.nn as nn
import torch.optim as optim
import matplotlib.pyplot as plt
import numpy as np
import torch.nn.functional as tf


# define model
class net(nn.Module):
    def __init__(self, n_feature, n_hidden, n_output):
        super(net, self).__init__()
        #self.poly = nn.Linear(6,1)
#         self.l1 = torch.nn.Linear(8, 6)
#         self.l2 = torch.nn.Linear(6, 5)
#         self.l3 = torch.nn.Linear(5, 4)
#         self.l4 = torch.nn.Linear(4, 3)
#         self.l5 = torch.nn.Linear(3, 2)
#         self.l6 = torch.nn.Linear(2, 1)
        self.hidden = torch.nn.Linear(8, 9)
        self.predict = torch.nn.Linear(9, 1)


    def forward(self, x):
        x = tf.relu(self.hidden(x))  #  function (linear value of hidden layer)
        x = self.predict(x)   # output
        return x
```

Multilayer Feed-Forward Neural Network: Train Model

- Here is criterion and optimizer
- Loss is too high

```
criterion = torch.nn.MSELoss(size_average=False)

optimizer = torch.optim.SGD(net.parameters(), lr=0.0001)
```

```
tensor(261035.9688, grad_fn=<SumBackward0>)
```

Adjust Loss: Research and Tuning Hyper-parameters

- Data standardized: standardization of data refers to scaling the data to make it fall into a specific interval
- Adjust optimizer
- Set different learning rates for different layers
- Create a new optimizer and adjust the learning rate

Prediction Results

LightGBM:

```
In [24]:  1  print('LightGBM results:',y_pred)

LightGBM results: [0.39616228 0.16217122 0.53179822 ... 0.16217122 0.21975678 0.16217122]
```

Random Forest:

```
7  print('prediction:',rf_pred_test)
8  print('RMSE',np.sqrt(metrics.mean_squared_error(y_test,rf_pred_test)))

prediction: [1.07289045 0.         2.00610394 ... 0.         0.         0.         ]
```

Multilayer Feed-forward Neural Network:

```
1  print ('MLP results:',prediction)

MLP results: tensor([[ 0.2365],
        [ 0.3104],
        [-0.1188],
        ...,
        [ 0.0150],
        [ 0.0648],
        [ 0.0783]], grad_fn=<ThAddmmBackward>)
```

What I Learn From Project

- LightGBM and Random Forest are Ensemble Learning
- LightGBM: Takes up less memory and has less complexity in data separation.
- Random Forest: it performs well and has high precision on large dataset; it is not easy to have over-fitting; it can get the order of importance of features
- Multi-layer feed forward neural network: The network essentially implements a mapping function from input to output, and mathematical theory has proven to have the ability to implement any complex nonlinear mapping.
- Adjust Parameters.

Advice to Companies

- Customer spending is concentrated on non-mobile platforms, but mobile platforms have great potential.
- With the rapid development of mobile phones, users will increase their consumption on mobile platforms.
- Due to the significant number of users in North America, some applications in other regions can be added.

Reference

- Kaggle:

https://www.kaggle.com/c/ga-customer-revenue-prediction

- LightGBM https://github.com/Microsoft/LightGBM/blob/master/examples/python-guide/sklearn_example.py

- Pytorch

https://github.com/MorvanZhou/PyTorch-Tutorial

Question ?

Columbian College of Arts & Sciences

Thank You