ctr

October 21, 2019

[1]:
```
# Importing Required Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

[2]:
```
# To avoid Depricated warnings
import warnings
import os
warnings.filterwarnings("ignore")
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
```

[3]:
```
# Function for parsing data from date column as date
def parser(x):
    return pd.datetime.strptime(x, '%Y%m%d')
```

[4]:
```
# Importing dataset as Pandas DataFrame
dataset = pd.read_csv("dataset.csv", parse_dates=[0], date_parser=parser,␣
 ↪header=None)
```

[5]:
```
# Droping NA values from Date, Market, Keyword
dataset = dataset[pd.isna(dataset[1]) == False]
dataset = dataset[pd.isna(dataset[0]) == False]
dataset = dataset[pd.isna(dataset[3]) == False]
```

[6]:
```
# Making Date column as pandas.DateTime
dataset.iloc[:, 0] = pd.to_datetime(dataset.iloc[:, 0])
```

[7]:
```
# Adding Day of Week number column into dataset
dataset[9] = dataset[0].apply(lambda x: x.dayofweek)
```

[8]:
```
# Adding Is_Holiday column into dataset for Holiday Date's
from pandas.tseries.holiday import USFederalHolidayCalendar as calendar
cal = calendar()
holidays = cal.holidays(start=dataset[0].min(), end=dataset[0].max())
dataset[10] = dataset[0].isin(holidays)
dataset.iloc[:, 10] = dataset.iloc[:, 10].astype("int")
```

[9]:
```
dataset.head(5)
```

[9]:
```
           0          1                                     2    3    4      5   \
0 2012-05-24  US-Market              secure online back up  0.0  0.0   0.00
1 2012-05-24  US-Market           agile management software  1.0  1.2  21.22
2 2012-05-24  US-Market                    crm for financial  0.0  0.0   0.00
3 2012-05-24  US-Market  disaster recovery planning for it  0.0  0.0   0.00
4 2012-05-24  US-Market                  tracking a vehicle  0.0  0.0   0.00

      6      7      8  9   10
0  0.00%    0.0   0.00  3   0
1  8.20%  260.0  25.45  3   0
2  0.00%    0.0   0.00  3   0
3  0.00%    0.0   0.00  3   0
4  0.00%    0.0   0.00  3   0
```

NLP Operations

[10]:
```
# Converting all keywords into lower case
dataset.iloc[:, 2] = dataset.iloc[:, 2].astype("str")
dataset[2] = dataset[2].apply(lambda x: x.lower())
```

[11]:
```
# Importing NLTK Libraries
import nltk
nltk.download('stopwords')
nltk.download('wordnet')
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()
ps = PorterStemmer()
```

```
[nltk_data] Downloading package stopwords to /home/ghost/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /home/ghost/nltk_data…
[nltk_data]   Package wordnet is already up-to-date!
```

[12]:
```
dataset.iloc[:, 2] = dataset.iloc[:, 2].apply(lambda x: x.split())
```

[13]:
```
def nlpo(x):
    ww = []
    for word in x:
        if not word in set(stopwords.words('english')):
          ww.append(lemmatizer.lemmatize(ps.stem(word)))
    return ww
```

```
dataset.iloc[:, 2] = dataset.iloc[:, 2].apply(lambda x: nlpo(x))
```

[14]:
```
dataset[2][:10]
```

[14]:
```
0          [secur, onlin, back]
1        [agil, manag, softwar]
2                [crm, financi]
3      [disast, recoveri, plan]
4               [track, vehicl]
5               [applic, cloud]
6       [project, manag, scrum]
7                [server, busi]
8    [android, applic, develop]
9       [android, app, develop]
Name: 2, dtype: object
```

[15]:
```
# joining the words again as sentence
dataset.iloc[:, 2] = dataset.iloc[:, 2].apply(lambda x: " ".join(x))
```

Model Training and Testing

[16]:
```
# Spliting the dataset
X = dataset.iloc[:, [1, 2, 4, 9, 10]].values
y = dataset.iloc[:, 5].values
```

[17]:
```
# Encoding Categorical data
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder_market = LabelEncoder()
X[:, 0] = labelencoder_market.fit_transform(X[:, 0])

labelencoder_keyword = LabelEncoder()
X[:, 1] = labelencoder_keyword.fit_transform(X[:, 1])
```

[18]:
```
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20,␣
↪random_state=0)
```

[19]:
```
# Fitting the RandomForest model to training dataset
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators=10, random_state=0, n_jobs=2)
regressor.fit(X_train, y_train)
```

[19]:
```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
                      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=10, n_jobs=2,
                       oob_score=False, random_state=0, verbose=0,
                       warm_start=False)
```

[20]:
```
y_pred = regressor.predict(X_test)
```

[21]:
```
# We can not apply confution matrix to continuous values
# so using cutoff method
cutoff = 0.7
y_pred_classes = np.zeros_like(y_pred)
y_pred_classes[y_pred > cutoff] = 1

y_test_classes = np.zeros_like(y_pred)
y_test_classes[y_test > cutoff] = 1
```

[22]:
```
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test_classes, y_pred_classes)
```

[29]:
```
# Accuracy with RandomForest Model
print(cm)
(30890+38504)/(30890+1073+137+38504)
```

```
[[30890  1073]
 [  137 38504]]
```

[29]:
```
0.9828621607840915
```

[25]:
```
import lightgbm as ltb
# fit a lightGBM model to the data
model = ltb.LGBMRegressor()
model.fit(X_train, y_train)
```

[25]:
```
LGBMRegressor(boosting_type='gbdt', class_weight=None, colsample_bytree=1.0,
              importance_type='split', learning_rate=0.1, max_depth=-1,
              min_child_samples=20, min_child_weight=0.001, min_split_gain=0.0,
              n_estimators=100, n_jobs=-1, num_leaves=31, objective=None,
              random_state=None, reg_alpha=0.0, reg_lambda=0.0, silent=True,
              subsample=1.0, subsample_for_bin=200000, subsample_freq=0)
```

[26]:
```
predicted_y = model.predict(X_test)
```

[27]:
```
y_pred_ltb = np.zeros_like(predicted_y)
y_pred_ltb[predicted_y > cutoff] = 1
y_test_ltb = np.zeros_like(predicted_y)
y_test_ltb[y_test > cutoff] = 1
cm_ltb = confusion_matrix(y_test_ltb, y_pred_ltb)
```

```
print(cm_ltb)
```

```
[[20198 11765]
 [ 2108 36533]]
```

[28]:
```
(20198+36533)/(20198+11765+2108+36533)
```

[28]:
```
0.8035097161633902
```