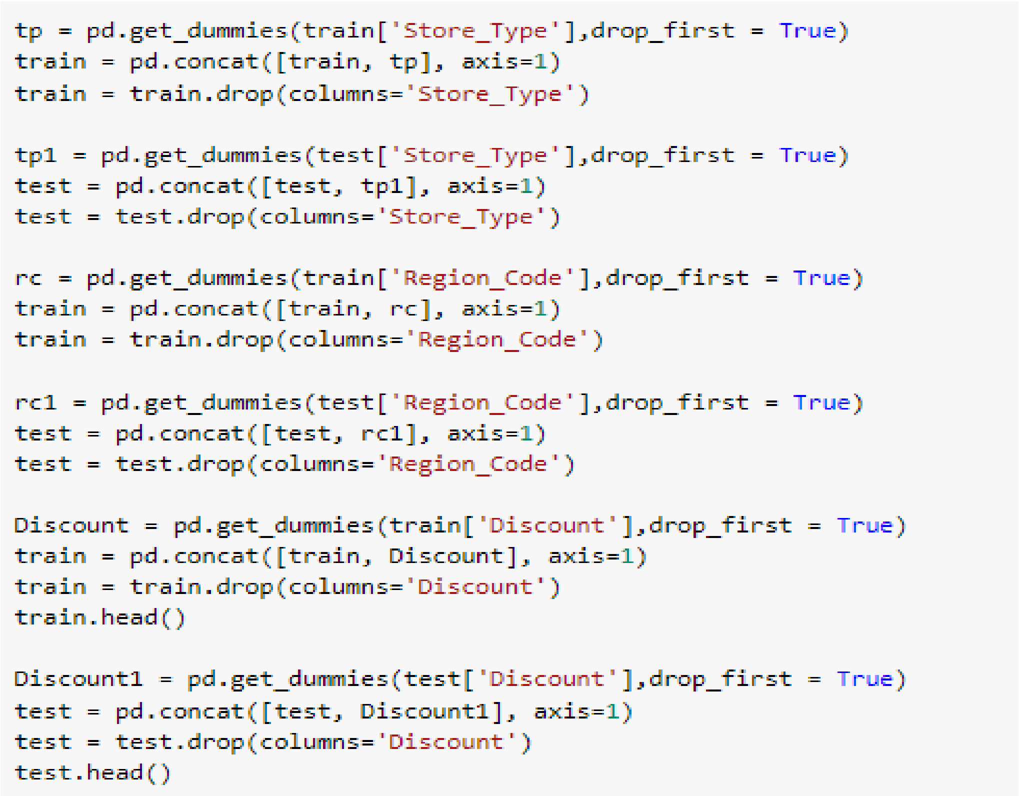
Approach (Jobthon) – 2021

By Rahul Dogra

1. I have used both LabelEncoding and OneHotEncoding. LabelEncoding I have used or ordinal features and OneHotEncoding for Nominal Data.

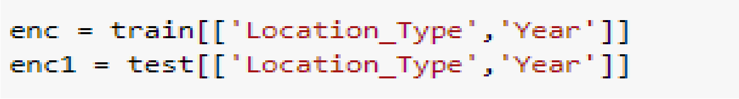
Using LabelEncoding might give weightage to numbers which are high in values so I have avoided it somewhere.

OneHotEncoding

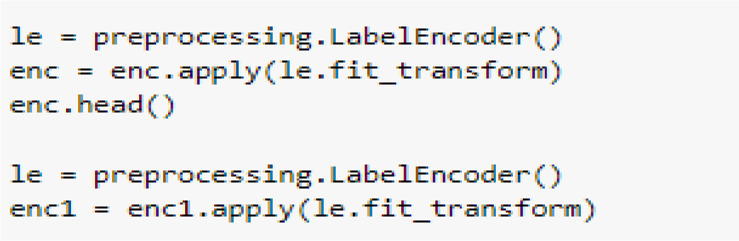


```
tp = pd.get_dummies(train['Store_Type'],drop_first = True)
train = pd.concat([train, tp], axis=1)
train = train.drop(columns='Store_Type')

tp1 = pd.get_dummies(test['Store_Type'],drop_first = True)
test = pd.concat([test, tp1], axis=1)
test = test.drop(columns='Store_Type')

rc = pd.get_dummies(train['Region_Code'],drop_first = True)
train = pd.concat([train, rc], axis=1)
train = train.drop(columns='Region_Code')

rc1 = pd.get_dummies(test['Region_Code'],drop_first = True)
test = pd.concat([test, rc1], axis=1)
test = test.drop(columns='Region_Code')

Discount = pd.get_dummies(train['Discount'],drop_first = True)
train = pd.concat([train, Discount], axis=1)
train = train.drop(columns='Discount')
train.head()

Discount1 = pd.get_dummies(test['Discount'],drop_first = True)
test = pd.concat([test, Discount1], axis=1)
test = test.drop(columns='Discount')
test.head()
```

LabelEncoding



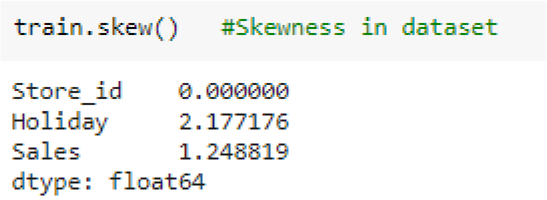
```
enc = train[['Location_Type','Year']]
enc1 = test[['Location_Type','Year']]
```



```
le = preprocessing.LabelEncoder()
enc = enc.apply(le.fit_transform)
enc.head()

le = preprocessing.LabelEncoder()
enc1 = enc1.apply(le.fit_transform)
```

3. Feature Engineering

I can really find some of skewness in the dataset:-

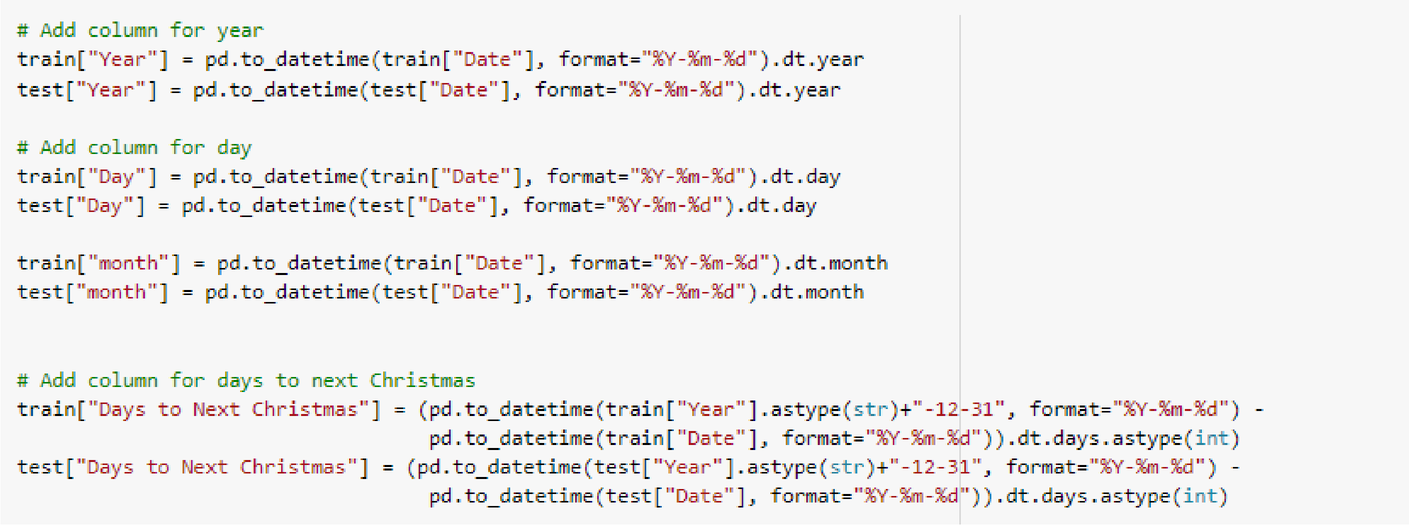


```
train.skew()   #Skewness in dataset
```

```
Store_id    0.000000
Holiday     2.177176
Sales       1.248819
dtype: float64
```

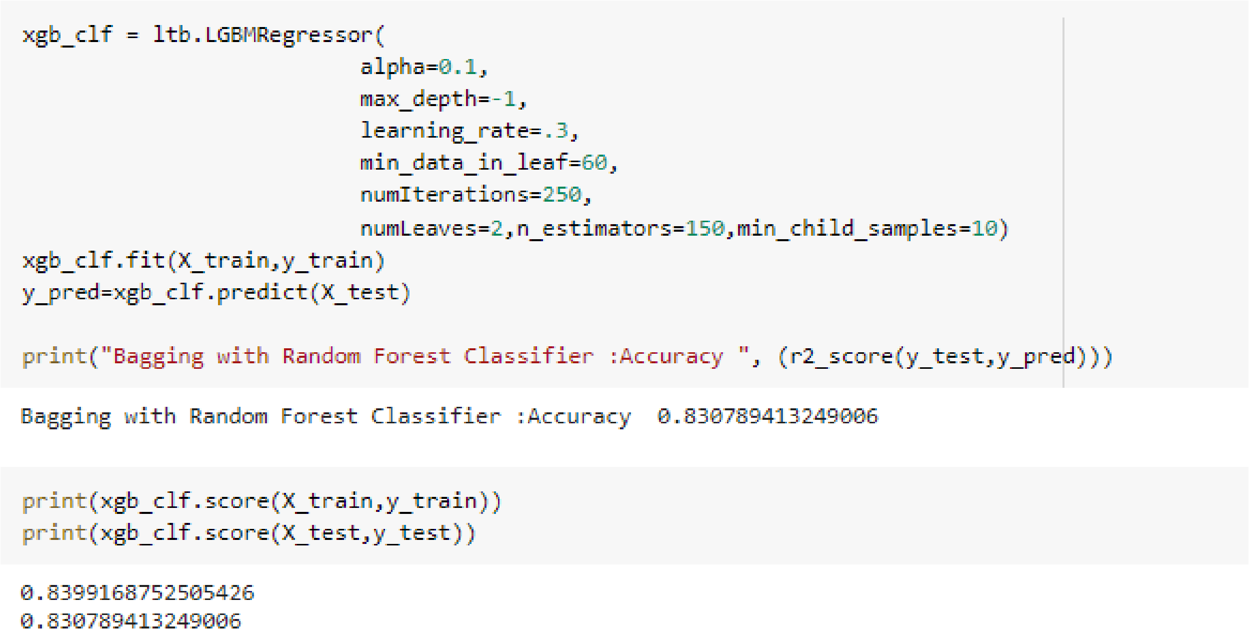
I tried to use Log Transformation to reduce the skewness on Holiday and Sales to perform well on this Regression Problem.

2. Splitted Date column into features to get more data to train



```
# Add column for year
train["Year"] = pd.to_datetime(train["Date"], format="%Y-%m-%d").dt.year
test["Year"] = pd.to_datetime(test["Date"], format="%Y-%m-%d").dt.year

# Add column for day
train["Day"] = pd.to_datetime(train["Date"], format="%Y-%m-%d").dt.day
test["Day"] = pd.to_datetime(test["Date"], format="%Y-%m-%d").dt.day

train["month"] = pd.to_datetime(train["Date"], format="%Y-%m-%d").dt.month
test["month"] = pd.to_datetime(test["Date"], format="%Y-%m-%d").dt.month


# Add column for days to next Christmas
train["Days to Next Christmas"] = (pd.to_datetime(train["Year"].astype(str)+"-12-31", format="%Y-%m-%d") -
                                   pd.to_datetime(train["Date"], format="%Y-%m-%d")).dt.days.astype(int)
test["Days to Next Christmas"] = (pd.to_datetime(test["Year"].astype(str)+"-12-31", format="%Y-%m-%d") -
                                   pd.to_datetime(test["Date"], format="%Y-%m-%d")).dt.days.astype(int)
```

3. Trained model using LightGBMRegressor:-



```
xgb_clf = ltb.LGBMRegressor(
                                alpha=0.1,
                                max_depth=-1,
                                learning_rate=.3,
                                min_data_in_leaf=60,
                                numIterations=250,
                                numLeaves=2,n_estimators=150,min_child_samples=10)
xgb_clf.fit(X_train,y_train)
y_pred=xgb_clf.predict(X_test)

print("Bagging with Random Forest Classifier :Accuracy ", (r2_score(y_test,y_pred)))
```

```
Bagging with Random Forest Classifier :Accuracy  0.830789413249006
```

```
print(xgb_clf.score(X_train,y_train))
print(xgb_clf.score(X_test,y_test))
```

```
0.8399168752505426
0.830789413249006
```

For better results you can use GridSearchCv to tune the hyperparamters of this model.

But I just got **217.25** on submission. You can improve more by using GridSearchCv.

Thank you,

Rahul Dogra

Total Experience – 4+ years