Luther Nicholaus
CIS 607
Unit 6 Assignment

Software Used

The software used for analysis is Python. The goal of the analysis is to build and train a classification model using historical sales records to predict if a product has been sold.

Dataset

The dataset used for analysis is titled "Historical Sales and Active Inventory Data Set". It is obtained from the UCI machine learning repository. It contains information about historical sales records, along with active inventory records. The historical and active records are distinguished in the column, FileType. The dataset holds 75997 historical sales records and 122922 active inventory records (James, 2016).

Attribute	Information
Order	Order serial number
File_Type	Distinguish whether the given product is a historical sales record or an active inventory record.
SKU_number	Unique product identifier
SoldFlag	Whether the product was sold or not. 1 = Sold 0 = Not sold
MarketingType	Marketing channel/type D - Direct S - Personal
ReleaseNumber	Release inventory
New_Release_Flag	Newly released inventory or not 1 - Yes 0 - No
StrengthFactor	Promotion intensity coefficient
PriceReg	Label price
ReleaseYear	Year of product release

ItemCount	Number of units
LowUserPrice	Lower individual product price
LowNetPrice	Lower price of the class from which the individual product belongs to

<u>Data Collection and Cleaning Procedures</u>

To achieve the predefined analysis goal, data collection and cleaning are performed first. A dataset that only holds historical sales records is extracted from the primary dataset using the Pandas library. Irrelevant columns for analysis are dropped. These include File_Type, Order, SKU_number, and SoldCount.

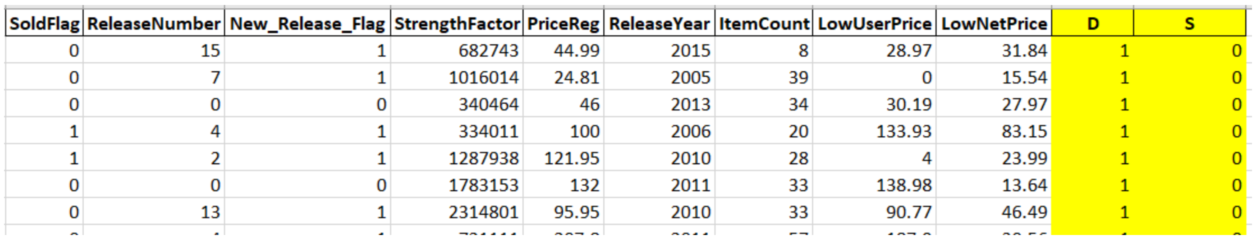
```
import pandas as pd
import numpy as np
import seaborn as sb
from sklearn.model_selection import train_test_split
from sklearn import tree

#read the dataset
df = pd.read_csv('C:/Users/18166/OneDrive/Documents/FileType_Historical.csv')

#drop irrelevant columns for analysis
df = df.drop(['File_Type', "Order", "Unnamed: 0", "SKU_number", "SoldCount"], axis=1)
```

To build a model, all attributes ought to contain numerical values. This means the non-numerical values in the MarketingType column are converted into numerical values with the help of dummy variable columns, as illustrated in Fig.1. In the D column, for instance, products that were marketed directly have the value of 1, and those that weren't have the value of 0. Once the later conversion process is completed, the MarketingType column containing the non-numerical values is dropped.

```
#convert the non-numeric columns to numeric
MarketingTypeDummy = pd.get_dummies(df['MarketingType'])
df = pd.concat((df, MarketingTypeDummy), axis=1)
df = df.drop(['MarketingType'], axis=1)
```



SoldFlag	ReleaseNumber	New_Release_Flag	StrengthFactor	PriceReg	ReleaseYear	ItemCount	LowUserPrice	LowNetPrice	D	S
0	15	1	682743	44.99	2015	8	28.97	31.84	1	0
0	7	1	1016014	24.81	2005	39	0	15.54	1	0
0	0	0	340464	46	2013	34	30.19	27.97	1	0
1	4	1	334011	100	2006	20	133.93	83.15	1	0
1	2	1	1287938	121.95	2010	28	4	23.99	1	0
0	0	0	1783153	132	2011	33	138.98	13.64	1	0
0	13	1	2314801	95.95	2010	33	90.77	46.49	1	0

Fig.1: With all attributes holding numerical values, analysis can resume

Description of the Analysis Procedure

The x and y data values are defined using the clean data frame. The x array values are from all columns, except the target column, SoldFlag. Recall the goal is to create a model to predict if a product has been sold or not. Based on the latter analytical objective, the y attribute is the SoldFlag column.

```
#separate the data frame into x and y data
x = df.values
y = df['SoldFlag'].values

#delete the survived column from x
x = np.delete(x,0,axis=1)
```

The code below splits the x and y values into 80%-to-20% chunks. The 80% chunk is used to train the classification model, while the 20% chunk is used to test and score the model.

```
#splitting the dataset into 80% Training and 20% Test
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2,random_state=0)
```

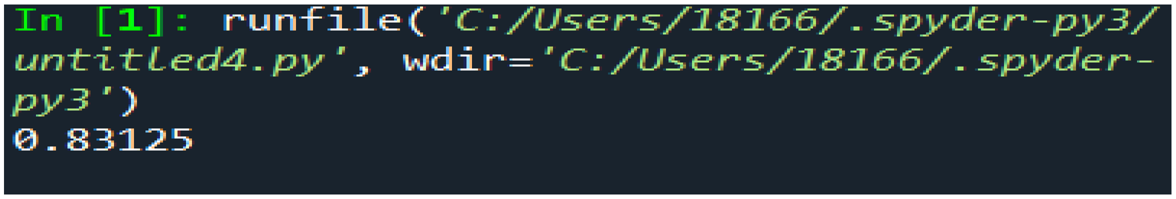
The code below builds and trains a decision tree classification using the x and y test values.

```
#Build and Train Decision Tree Classifier
dt_clf =tree.DecisionTreeClassifier(max_depth=5)

dt_clf.fit(x_train, y_train)
```

Lastly, the classifier is tested and scored for prediction using the train sample of x and y values. The outcome is displayed in Fig.2.

```
#Prediction
score = dt_clf.score(x_test, y_test)
print(score)
```



```
In [1]: runfile('C:/Users/18166/.spyder-py3/
untitled4.py', wdir='C:/Users/18166/.spyder-
py3')
0.83125
```

Fig.2: Classification model score.

Conclusion

The accuracy of the model to predict if the product is sold or not is 83.13%. This score is based on the mean accuracy of the given test labels and data. With an 83.13% score, the decision tree classifier appears to be a solid predictor of whether a product has been sold or not.

References

James, S. (2016). Historical Sales and Active Inventory Data Set. *Kaggle.com*. Retrieved from https://www.kaggle.com/flenderson/sales-analysis/metadata