Homework 9 (Optional)

author: Anthony Martinez \ CS 4375

Load The Data

In [140…

```
import pandas as pd
df0 = pd.read_csv('data/blackfriday.csv')
```

Data Cleaning

Link to Dataset: https://www.kaggle.com/sdolezel/black-friday?select=train.csv Data Cleaning Operations:

1. Count NAs, if any
2. Remove NAs
3. Convert qualitative data into factors

In [141…

```
# 1. Count NAs
df0.isnull().sum()
# Product_Category_2 and Product_Category_3 are the only columns with NA

# 2. Remove NAs
df = df0.drop(columns=['Product_Category_2', 'Product_Category_3'])

# confirm removal of columns
df.isnull().sum()

# 3. Convert qualitative data into factors
df.dtypes


'''
The following columns will be converted to category type using cat codes
- Gender
- Age
- Occupation
- Stay_In_Current_City_Years
- City_Category
- Marital Status
- Product_Category
'''

df1 = df.copy()
df1.Gender = df1.Gender.astype('category').cat.codes
df1.Age = df1.Age.astype('category').cat.codes
df1.Occupation = df1.Occupation.astype('category').cat.codes
df1.City_Category = df1.City_Category.astype('category').cat.codes
df1.Stay_In_Current_City_Years = df1.Stay_In_Current_City_Years.astype('category
df1.Marital_Status = df1.Marital_Status.astype('category').cat.codes
df1.Product_Category_1 = df1.Product_Category_1.astype('category').cat.codes
```

```
# Checking that all columns were set to category type
df1.dtypes
```

Out[141…

```
User_ID                        int64
Product_ID                    object
Gender                          int8
Age                             int8
Occupation                      int8
City_Category                   int8
Stay_In_Current_City_Years      int8
Marital_Status                  int8
Product_Category_1              int8
Purchase                       int64
dtype: object
```

Data Exploration

- use at least 5 R functions for data exploration
- create at least 3 informative R graphs for data exploration

In [142…

```
'''
5 R functions
1. head
2. info
3. info
4 columns
5. mean(purchase)
'''

# same as R's head() function
print(df1.head())

# describe() is similar to the summary() in R
print('\n',df1.describe())

# info() is similar to R's str()
print('\n', df1.info())

# .columns is similar to colnames(df) in R
print('\n', df1.columns)

print('Mean of Purchase column (target column)', df1["Purchase"].mean())
```

```
   User_ID Product_ID  Gender  Age  Occupation  City_Category  \
0  1000001  P00069042       0    0          10              0
1  1000001  P00248942       0    0          10              0
2  1000001  P00087842       0    0          10              0
3  1000001  P00085442       0    0          10              0
4  1000002  P00285442       1    6          16              2

   Stay_In_Current_City_Years  Marital_Status  Product_Category_1  Purchase
0                           2               0                   2      8370
1                           2               0                   0     15200
2                           2               0                  11      1422
3                           2               0                  11      1057
4                           4               0                   7      7969

                User_ID          Gender           Age      Occupation  \
```

```
count  5.500680e+05  550068.000000  550068.000000  550068.000000
mean   1.003029e+06       0.753105       2.496430       8.076707
std    1.727592e+03       0.431205       1.353632       6.522660
min    1.000001e+06       0.000000       0.000000       0.000000
25%    1.001516e+06       1.000000       2.000000       2.000000
50%    1.003077e+06       1.000000       2.000000       7.000000
75%    1.004478e+06       1.000000       3.000000      14.000000
max    1.006040e+06       1.000000       6.000000      20.000000

       City_Category  Stay_In_Current_City_Years  Marital_Status  \
count  550068.000000               550068.000000   550068.000000
mean        1.042640                    1.858418        0.409653
std         0.760211                    1.289443        0.491770
min         0.000000                    0.000000        0.000000
25%         0.000000                    1.000000        0.000000
50%         1.000000                    2.000000        0.000000
75%         2.000000                    3.000000        1.000000
max         2.000000                    4.000000        1.000000

       Product_Category_1       Purchase
count       550068.000000  550068.000000
mean             4.404270    9263.968713
std              3.936211    5023.065394
min              0.000000      12.000000
25%              0.000000    5823.000000
50%              4.000000    8047.000000
75%              7.000000   12054.000000
max             19.000000   23961.000000
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 10 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   User_ID                     550068 non-null  int64
 1   Product_ID                  550068 non-null  object
 2   Gender                      550068 non-null  int8
 3   Age                         550068 non-null  int8
 4   Occupation                  550068 non-null  int8
 5   City_Category               550068 non-null  int8
 6   Stay_In_Current_City_Years  550068 non-null  int8
 7   Marital_Status              550068 non-null  int8
 8   Product_Category_1          550068 non-null  int8
 9   Purchase                    550068 non-null  int64
dtypes: int64(2), int8(7), object(1)
memory usage: 16.3+ MB

 None

 Index(['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Category',
       'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category_1',
       'Purchase'],
      dtype='object')
Mean of Purchase column (target column) 9263.968712959126
```

Data Exploration: Graphs

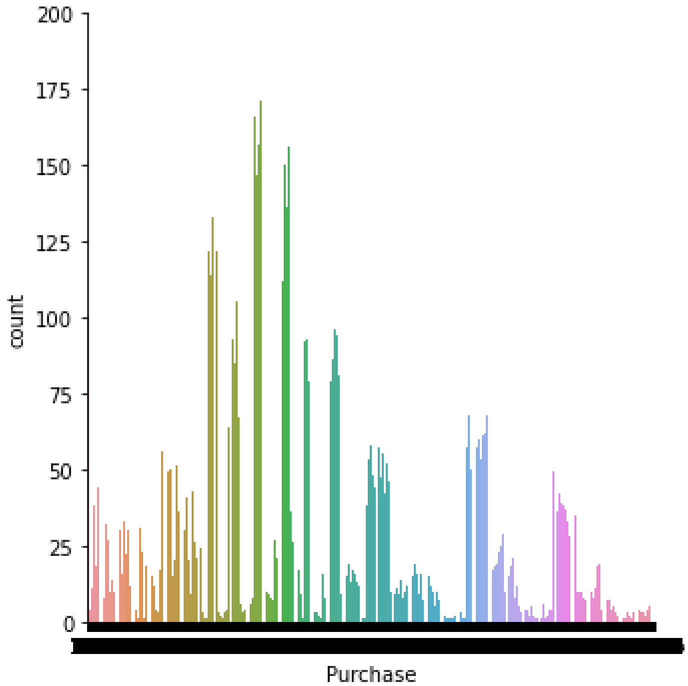
In [143…

```python
import seaborn as sb
```

```
# 1. seaborn catplot on the purchase column
sb.catplot(x='Purchase', kind='count', data=df1)
```

Out[143…

```
<seaborn.axisgrid.FacetGrid at 0x7fb456e289d0>
```



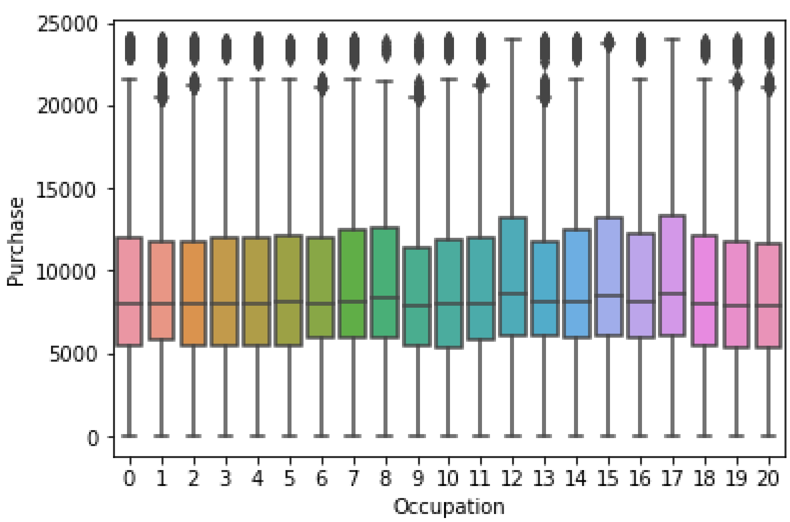
In [144…

```
import seaborn as sb

# 2. seaborn boxplot with Occupation on the x axis and Purchase on the y axis
sb.boxplot(x='Occupation', y='Purchase', data=df1)
```

Out[144…

```
<AxesSubplot:xlabel='Occupation', ylabel='Purchase'>
```



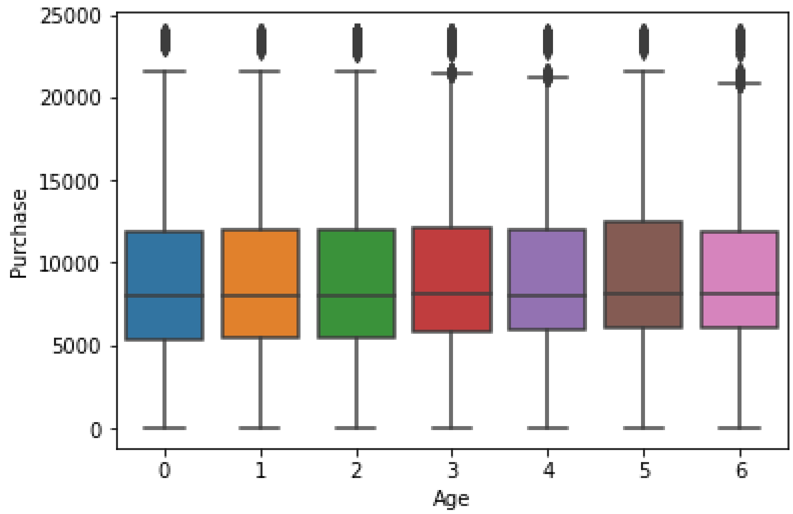
In [162…

```
import seaborn as sb

# 3. seaborn boxplot with Age on the x axis and Purchase on the y axis
sb.boxplot(x='Age', y='Purchase', data=df1)
```

Out[162…

```
<AxesSubplot:xlabel='Age', ylabel='Purchase'>
```



Train/Test Split (80/20)

In [145…

```
from sklearn.model_selection import train_test_split

# Predicting Purchase from Age, Occupation, City_Category, and Product_Category_
X = df1.loc[:,['Age', 'Occupation', 'City_Category', 'Product_Category_1']]
y = df1.Purchase

# 80/20 split, using seed 1234 for repeatability
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_

# printing sizes of train and test
print('train size:', X_train.shape)
print('test size:', X_test.shape)
```

```
train size: (440054, 4)
test size: (110014, 4)
```

Algorithm 1: Linear Regression

In [153…

```
from sklearn.linear_model import LinearRegression

linreg = LinearRegression()
linreg.fit(X_train, y_train)

# make predictions
y_pred = linreg.predict(X_test)

# evaluation
from sklearn.metrics import mean_squared_error, r2_score
print('mse=', mean_squared_error(y_test, y_pred))
print('correlation=', r2_score(y_test, y_pred))
```

```
mse= 22126347.338500306
correlation= 0.12262488554739837
```

Algorithm 2: Decision Tree

In [158…

```
from sklearn.tree import DecisionTreeRegressor
```

```
from sklearn import tree

regressor = DecisionTreeRegressor()

regressor.fit(X_train, y_train)
pred_dt = regressor.predict(X_test)

# evaluation
from sklearn.metrics import mean_squared_error, r2_score
print('mse=', mean_squared_error(y_test, pred_dt))
print('correlation=', r2_score(y_test, pred_dt))
```

```
mse= 8999419.148505524
correlation= 0.6431464134304717
```

Algorithm 3: KNN Regression

In [160…

```
# train the algorithm
from sklearn.neighbors import KNeighborsRegressor
regressor2 = KNeighborsRegressor()
regressor2.fit(X_train, y_train)

# make predictions
pred_knn = regressor2.predict(X_test)

# evaluation
from sklearn.metrics import mean_squared_error, r2_score
print('mse=', mean_squared_error(y_test, y_pred))
print('correlation=', r2_score(y_test, y_pred))
```

```
mse= 10476668.805379316
correlation= 0.5845690953152732
```

Analysis:

The linear regression algorithm got a mse of 22126347.34 and a correlation= 0.1226 The Decision tree regression algorithm got a mse of 8999419.15 and a correlation of 0.64 The kNN regression algorithm got a mse mse of 10476668.81 and a correlation of 0.585 Rank:

1. Decision Tree Regression
2. kNN Regression
3. Linear Regression (multiple)

Decision Tree Regression did the best out of all three algorithms. KNN preformed the 2nd best. This is probably because the algorithm uses feature similarity to make predictions.The reason linear regression probably did poorly is because of its high bias. The predictor data varies wildly. Linear Regression will find a line, even if it makes no sense.

Machine Learning in R v.s Python

Coming into this course I thought I would enjoy using Python more than R. I was a little upset when I found out this course is taught in R and not Python. Python is my favorite language to build projects and it still is. However, when it comes to Machine Learning, data cleaning, and data exploration R is more intuitive. Another important factor is R-Studio. R-Studio is a great IDE for ML and for working with data. I tried to find an IDE similar to R-Studio, but for Python. The best option I came across was DataSpell. The IDE is pretty great but I think it's safe to say that R-Studio is a better option. I think it is also important to note that we did a lot more assignments using R than in comparison to Python, so it's not much of a fair comparison. One thing I have noticed, especially towards the end of this course, is that it seems Machine Learning with Python seems to have more powerful tools for Deep Learning such as Keras and TensorFlow. It also seems that these libraries are in high demand when it comes to the current job posting. It is for this last reason that I have decided to continue my Deep Learning and Machine Learning studies using Python even though my personal opinion is that R/R-Studio is more user friendly than Python.