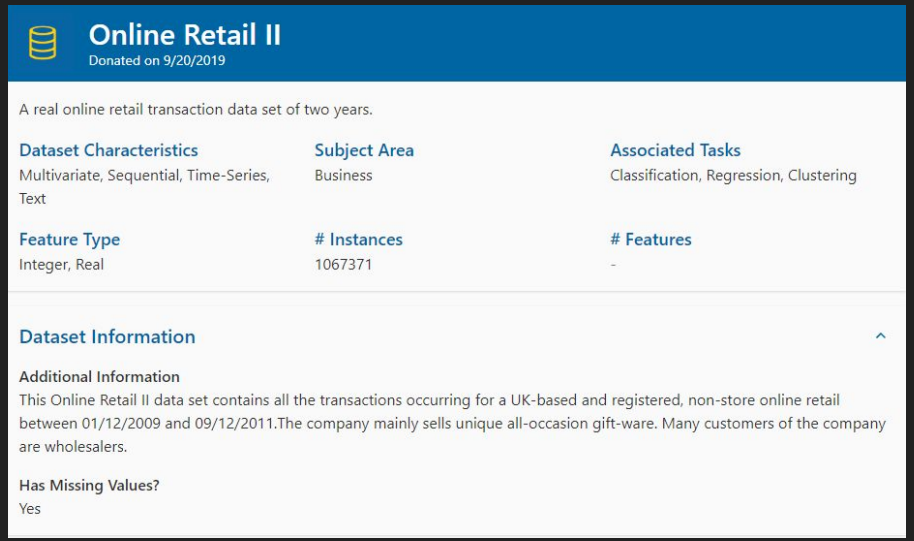
Data 301 Project

Analyzing an Online Retail Store

Yohan Sofian and Connor Gamba

Data Set Used



Online Retail II
Donated on 9/20/2019

A real online retail transaction data set of two years.

Dataset Characteristics	Subject Area	Associated Tasks
Multivariate, Sequential, Time-Series, Text	Business	Classification, Regression, Clustering
Feature Type	**# Instances**	**# Features**
Integer, Real	1067371	-

Dataset Information

Additional Information
This Online Retail II data set contains all the transactions occurring for a UK-based and registered, non-store online retail between 01/12/2009 and 09/12/2011.The company mainly sells unique all-occasion gift-ware. Many customers of the company are wholesalers.

Has Missing Values?
Yes

Variables Included:

-Invoice (str)
*(Values starting with c indicate cancellation)
-StockCode (str)
-Description (str)
-Quantity (int)
-InvoiceDate (str)
*(Converted to datetime)
-Price (float)
*Prices in British Pounds
-Customer ID (float)
-Country (str)

Chen,Daqing. (2019). Online Retail II. UCI Machine Learning Repository. https://doi.org/10.24432/C5CG6D .

Feature Engineered Variables

Time Variables (int)

- Month
- Day
- Year
- Hour
- Minute
- Quarter

Str Variables (str)

- CustomerID_str
- Invoice_str
- StockCode_str

Purchase Variables

- TotalAmount (float)
- MonthlyPurchaseQuantity (int)
- MonthlyPurchaseAmount (float)
- YearlyPurchaseQuantity (int)
- YearlyPurchaseAmount (float)

Questions Involving Simple Analytical Analysis

Questions that involve simple analytical analysis are as follows:

- What item is the best seller in every month based on the dataset?
- What item is the best seller in every quarter based on the dataset?
- Which region is this business most successful in? Calculate the total sales for each country and find the biggest sales.
- Which region is this business least successful in? Calculate the total sales for each country and find the lowest sales.
- What is the largest transaction on an invoice based on the dataset?
- What is the mean revenue of this business per month?
- Calculate total revenue for this business for each month based on the dataset
- Calculate the average number of transactions per month for this business.
- Calculate the average total sales on the invoice for this business.
- Predict the average revenue per month by multiplying the average number of transactions (invoices) with the average total sales per invoice.
- What is the best selling product for each country?
- What is the product that generates the most revenue for each country?

What item is the best seller in every month based on the dataset?

```
monthly_sales = df.groupby(['Month', 'StockCode'])['Quantity'].sum().reset_index()
best_sellers = monthly_sales.loc[monthly_sales.groupby('Month')['Quantity'].idxmax()]
best_sellers = best_sellers.merge(df[['StockCode', 'Description']], on='StockCode', how='left')
for month in best_sellers['Month'].unique():
    item_info = best_sellers[best_sellers['Month'] == month].iloc[0]
    print(f"In month {month}, the best-selling item is '{item_info['Description']}' (StockCode: {item_info['StockCode']}) with a quantity of {item_info['Quantity']}.")
```

```
In month 1, the best-selling item is 'JAZZ HEARTS MEMO PAD' (StockCode: 20993) with a quantity of 9500.
In month 2, the best-selling item is 'BLACK AND WHITE PAISLEY FLOWER MUG' (StockCode: 37410) with a quantity of 19249.
In month 3, the best-selling item is 'SET/6 WOODLAND PAPER PLATES' (StockCode: 21091) with a quantity of 13107.
In month 4, the best-selling item is 'PACK OF 72 RETRO SPOT CAKE CASES' (StockCode: 21212) with a quantity of 5365.
In month 5, the best-selling item is 'FLAG OF ST GEORGE CAR FLAG' (StockCode: 84016) with a quantity of 11097.
In month 6, the best-selling item is 'WORLD WAR 2 GLIDERS ASSTD DESIGNS' (StockCode: 84077) with a quantity of 5515.
In month 7, the best-selling item is 'PACK OF 72 RETRO SPOT CAKE CASES' (StockCode: 21212) with a quantity of 4181.
In month 8, the best-selling item is 'SET/6 FRUIT SALAD PAPER CUPS' (StockCode: 21088) with a quantity of 7131.
In month 9, the best-selling item is 'BROCADE RING PURSE ' (StockCode: 17003) with a quantity of 13895.
In month 10, the best-selling item is 'PACK OF 72 RETRO SPOT CAKE CASES' (StockCode: 21212) with a quantity of 6544.
In month 11, the best-selling item is 'ROTATING SILVER ANGELS T-LIGHT HLDR' (StockCode: 84347) with a quantity of 12445.
In month 12, the best-selling item is 'WHITE HANGING HEART T-LIGHT HOLDER' (StockCode: 85123A) with a quantity of 8034.
```

What item is the best seller in every quarter based on the dataset?

```
[6] df['Quarter'] = df['InvoiceDate'].dt.quarter
    quarterly_sales = df.groupby(['Quarter', 'StockCode'])['Quantity'].sum().reset_index()
    best_sellers = quarterly_sales.loc[quarterly_sales.groupby('Quarter')['Quantity'].idxmax()]
    best_sellers = best_sellers.merge(df[['StockCode', 'Description']], on='StockCode', how='left')
    best_sellers = best_sellers[['Quarter', 'StockCode', 'Description', 'Quantity']]
    for quarter in best_sellers['Quarter'].unique():
        item_info = best_sellers[best_sellers['Quarter'] == quarter].iloc[0]
        print(f"In quarter {quarter}, the best-selling item is '{item_info['Description']}' (StockCode: {item_info['StockCode']}) with a quantity of {item_info['Quantity']}.")
```

```
In quarter 1, the best-selling item is 'BLACK AND WHITE PAISLEY FLOWER MUG' (StockCode: 37410) with a quantity of 24988.
In quarter 2, the best-selling item is 'FLAG OF ST GEORGE CAR FLAG' (StockCode: 84016) with a quantity of 17281.
In quarter 3, the best-selling item is 'BROCADE RING PURSE ' (StockCode: 17003) with a quantity of 17030.
In quarter 4, the best-selling item is 'WORLD WAR 2 GLIDERS ASSTD DESIGNS' (StockCode: 84077) with a quantity of 21478.
```

Which region is this business most successful in? Calculate the total sales for each country and find the biggest sales.

```
country_sales = df.groupby('Country')['Quantity'].sum().reset_index()
most_successful_region = country_sales.loc[country_sales['Quantity'].idxmax()]
print("The most successful region for the business is:", most_successful_region['Country'])
```

```
The most successful region for the business is: United Kingdom
```

+ Code + Text

Which region is this business least successful in? Calculate the total sales for each country and find the lowest sales.

```
[8] most_successful_region = country_sales.loc[country_sales['Quantity'].idxmin()]
    print("The least successful region for the business is:", most_successful_region['Country'])
```

```
The least successful region for the business is: Nigeria
```

What is the largest transaction on an invoice based on the dataset?

```
df['TotalAmount'] = df['Quantity'] * df['Price']
invoice_totals = df.groupby('Invoice')['TotalAmount'].sum().reset_index()
largest_transaction = invoice_totals.loc[invoice_totals['TotalAmount'].idxmax()]
print("The largest transaction on an invoice is: Total Amount =", largest_transaction['TotalAmount'],
      "Invoice Number =", largest_transaction['Invoice'])
```

```
The largest transaction on an invoice is: Total Amount = 49844.990000000005 Invoice Number = 533027
```

What is the mean revenue of this business per month?

```
[10] mean_revenue_per_month = df.groupby('Month')['TotalAmount'].sum().mean()
     print("The mean revenue of the business per month is:", mean_revenue_per_month)
```

```
The mean revenue of the business per month is: 794957.0528333333
```

Calculate total revenue for this business for each month based on the dataset

```
monthly_revenue = df.groupby('Month')['TotalAmount'].sum()
for month, revenue in monthly_revenue.items():
    print(f"Month {month}: Total Revenue = {revenue}")
```

```
Month 1: Total Revenue = 624032.892
Month 2: Total Revenue = 533091.426
Month 3: Total Revenue = 765848.761
Month 4: Total Revenue = 590580.432
Month 5: Total Revenue = 615322.83
Month 6: Total Revenue = 679786.61
Month 7: Total Revenue = 575236.36
Month 8: Total Revenue = 656776.34
Month 9: Total Revenue = 853650.431
Month 10: Total Revenue = 1045168.35
Month 11: Total Revenue = 1422654.642
Month 12: Total Revenue = 1177335.56
```

Calculate the average number of transactions per month for this business.

```
[12] monthly_transactions = df.groupby('Month')['Invoice'].nunique()
     average_transactions_per_month = monthly_transactions.mean()
     print("The average number of transactions per month for this business is:", average_transactions_per_month)
```

```
The average number of transactions per month for this business is: 2401.3333333333335
```

Calculate the average total sales on the invoice for this business.

```
[13] invoice_total_sales = df.groupby('Invoice')['TotalAmount'].sum()
     average_total_sales_per_invoice = invoice_total_sales.mean()
     print("The average total sales on each invoice for this business is:", average_total_sales_per_invoice)
```

```
The average total sales on each invoice for this business is: 331.0481896862854
```

Predict the average revenue per month by multiplying the average number of transactions (invoices) with the average total sales per invoice.

```
[14] print(average_transactions_per_month * average_total_sales_per_invoice)
```

```
794957.0528333334
```

Predict the average revenue per month by multiplying the average number of transactions (invoices) with the average total sales per invoice.

```
print(average_transactions_per_month * average_total_sales_per_invoice)
```

```
794957.0528333334
```

What is the best selling product for each country?

```
country_sales = df.groupby(['Country', 'StockCode'])['Quantity'].sum().reset_index()
country_best_seller = country_sales.loc[country_sales.groupby('Country')['Quantity'].idxmax()]
country_best_seller = country_best_seller.merge(df[['StockCode', 'Description']], on='StockCode', how='left')
country_best_seller.head()
```

```
for country in country_best_seller['Country'].unique():
  item_info = country_best_seller[country_best_seller['Country'] == country].iloc[0]
  print(f"For {country}, the best-selling item is '{item_info['Description']}' (StockCode: {item_info['StockCode']}) with a quantity of {item_info['Quantity']}.")
```

```
For Australia, the best-selling item is '72 CAKE CASES DOLLY GIRL DESIGN' (StockCode: 22951) with a quantity of 504.
For Austria, the best-selling item is 'MINI HIGHLIGHTER PENS' (StockCode: 16033) with a quantity of 120.
For Bahrain, the best-selling item is 'WHITE TALL PORCELAIN T-LIGHT HOLDER' (StockCode: 18097C) with a quantity of 102.
For Belgium, the best-selling item is 'PACK OF 72 RETRO SPOT CAKE CASES' (StockCode: 21212) with a quantity of 336.
For Bermuda, the best-selling item is 'GIRLS ALPHABET IRON ON PATCHES ' (StockCode: 84568) with a quantity of 1152.
For Brazil, the best-selling item is 'DRAGONS BLOOD INCENSE' (StockCode: 17084P) with a quantity of 25.
For Canada, the best-selling item is 'JAZZ HEARTS ADDRESS BOOK' (StockCode: 20996) with a quantity of 24.
For Channel Islands, the best-selling item is 'JUMBO BAG PINK WITH WHITE SPOTS' (StockCode: 22386) with a quantity of 330.
For Cyprus, the best-selling item is 'BLUE SCANDINAVIAN PAISLEY WRAP' (StockCode: 22051) with a quantity of 200.
For Denmark, the best-selling item is 'BLACK AND WHITE PAISLEY FLOWER MUG' (StockCode: 37410) with a quantity of 25164.
For EIRE, the best-selling item is 'PACK OF 72 RETRO SPOT CAKE CASES' (StockCode: 21212) with a quantity of 4224.
For Finland, the best-selling item is 'PINK 3 PIECE MINI DOTS CUTLERY SET' (StockCode: 84997D) with a quantity of 156.
For France, the best-selling item is 'RED TOADSTOOL LED NIGHT LIGHT' (StockCode: 21731) with a quantity of 1471.
For Germany, the best-selling item is 'WOODLAND CHARLOTTE BAG' (StockCode: 20719) with a quantity of 1585.
For Greece, the best-selling item is 'WHITE HANGING HEART T-LIGHT HOLDER' (StockCode: 85123A) with a quantity of 160.
For Hong Kong, the best-selling item is 'PLASTERS IN TIN SPACEBOY' (StockCode: 22551) with a quantity of 108.
For Iceland, the best-selling item is 'MINI PAINT SET VINTAGE ' (StockCode: 22492) with a quantity of 36.
For Israel, the best-selling item is 'SALLE DE BAIN HOOK' (StockCode: 21272) with a quantity of 36.
For Italy, the best-selling item is 'INFLATABLE POLITICAL GLOBE ' (StockCode: 10002) with a quantity of 360.
For Japan, the best-selling item is 'ROUND SNACK BOXES ,SET 4, FRUITS ' (StockCode: 22328) with a quantity of 1488.
For Korea, the best-selling item is 'TROPICAL  HONEYCOMB PAPER GARLAND ' (StockCode: 21201) with a quantity of 48.
For Lebanon, the best-selling item is 'PLASTERS IN TIN SKULLS' (StockCode: 22553) with a quantity of 12.
For Lithuania, the best-selling item is 'SILVER MUG BONE CHINA TREE OF LIFE' (StockCode: 22306) with a quantity of 72.
For Malta, the best-selling item is 'VICTORIAN SEWING KIT' (StockCode: 85178) with a quantity of 144.
For Netherlands, the best-selling item is 'FOLKART ZINC HEART CHRISTMAS DEC' (StockCode: 35961) with a quantity of 5425.
For Nigeria, the best-selling item is 'EMPIRE GIFT WRAP' (StockCode: 22047) with a quantity of 25.
For Norway, the best-selling item is 'PINK HEART SHAPE EGG FRYING PAN' (StockCode: 84050) with a quantity of 194.
For Poland, the best-selling item is 'STRAWBERRY CERAMIC TRINKET BOX' (StockCode: 21232) with a quantity of 120.
For Portugal, the best-selling item is 'LUNCH BAG RED SPOTTY' (StockCode: 20725) with a quantity of 260.
For RSA, the best-selling item is 'PINK  SPOTTY CUP' (StockCode: 21239) with a quantity of 104.
For Singapore, the best-selling item is 'WHITE HANGING HEART T-LIGHT HOLDER' (StockCode: 85123A) with a quantity of 96.
For Spain, the best-selling item is 'BLUE 3 PIECE MINI DOTS CUTLERY SET' (StockCode: 84997C) with a quantity of 648.
For Sweden, the best-selling item is 'MINI HIGHLIGHTER PENS' (StockCode: 16033) with a quantity of 5760.
For Switzerland, the best-selling item is 'GIRLS ALPHABET IRON ON PATCHES ' (StockCode: 84568) with a quantity of 288.
For Thailand, the best-selling item is 'ENVELOPE 50 ROMANTIC IMAGES' (StockCode: 85017A) with a quantity of 288.
For USA, the best-selling item is 'TOAST ITS - I LOVE YOU ' (StockCode: 21355) with a quantity of 408.
For United Arab Emirates, the best-selling item is 'GIRLS ALPHABET IRON ON PATCHES ' (StockCode: 84568) with a quantity of 288.
For United Kingdom, the best-selling item is 'WHITE HANGING HEART T-LIGHT HOLDER' (StockCode: 85123A) with a quantity of 51755.
For Unspecified, the best-selling item is 'MINI HIGHLIGHTER PENS' (StockCode: 16033) with a quantity of 120.
For West Indies, the best-selling item is 'BROCADE RING PURSE ' (StockCode: 17003) with a quantity of 36.
```

What is the product that generates the most revenue for each country?

```
[20] country_product_revenue = df.groupby(['Country', 'Description'])['TotalAmount'].sum()
     best_country_product = country_product_revenue.groupby('Country').idxmax()
     country_winning_product = country_product_revenue.loc[best_country_product]
```

```
country_product_revenue = df.groupby(['Country', 'StockCode'])['TotalAmount'].sum().reset_index()
best_country_product = country_product_revenue.loc[country_product_revenue.groupby('Country')['TotalAmount'].idxmax()]
best_country_product = best_country_product.merge(df[['StockCode', 'Description']], on='StockCode', how='left')

for country in best_country_product['Country'].unique():
  item_info = best_country_product[best_country_product['Country'] == country].iloc[0]
  print(f"For {country}, the product that generates the most revenue is '{item_info['Description']}' (StockCode: {item_info['StockCode']}) with total sales of ${item_info['TotalAmount']}.")
```

```
For Australia, the product that generates the most revenue is 'REGENCY CAKESTAND 3 TIER' (StockCode: 22423) with total sales of $927.0.
For Austria, the product that generates the most revenue is 'POSTAGE' (StockCode: POST) with total sales of $1520.0.
For Bahrain, the product that generates the most revenue is 'WHITE TALL PORCELAIN T-LIGHT HOLDER' (StockCode: 18097C) with total sales of $202.5.
For Belgium, the product that generates the most revenue is 'POSTAGE' (StockCode: POST) with total sales of $2398.0.
For Bermuda, the product that generates the most revenue is 'GIRLS ALPHABET IRON ON PATCHES ' (StockCode: 84568) with total sales of $241.92.
For Brazil, the product that generates the most revenue is 'FRENCH PAISLEY CUSHION COVER ' (StockCode: 20839) with total sales of $17.700000000000003.
For Canada, the product that generates the most revenue is 'RETRO SPOT SMALL TUBE MATCHES' (StockCode: 21584) with total sales of $33.0.
For Channel Islands, the product that generates the most revenue is 'AFGHAN SLIPPER SOCK PAIR' (StockCode: 51008) with total sales of $885.0.
For Cyprus, the product that generates the most revenue is 'REGENCY CAKESTAND 3 TIER' (StockCode: 22423) with total sales of $567.15.
For Denmark, the product that generates the most revenue is 'SMALL FAIRY CAKE FRIDGE MAGNETS' (StockCode: 85220) with total sales of $2836.7999999999997.
For EIRE, the product that generates the most revenue is 'REGENCY CAKESTAND 3 TIER' (StockCode: 22423) with total sales of $7524.9.
For Finland, the product that generates the most revenue is 'PINK 3 PIECE MINI DOTS CUTLERY SET' (StockCode: 84997D) with total sales of $533.1600000000001.
For France, the product that generates the most revenue is 'POSTAGE' (StockCode: POST) with total sales of $9442.0.
For Germany, the product that generates the most revenue is 'POSTAGE' (StockCode: POST) with total sales of $17803.0.
For Greece, the product that generates the most revenue is 'WHITE HANGING HEART T-LIGHT HOLDER' (StockCode: 85123A) with total sales of $408.0.
For Hong Kong, the product that generates the most revenue is 'RED 3 PIECE MINI DOTS CUTLERY SET' (StockCode: 84997B) with total sales of $244.08.
For Iceland, the product that generates the most revenue is '3D DOG PICTURE PLAYING CARDS' (StockCode: 84558A) with total sales of $88.50000000000001.
For Israel, the product that generates the most revenue is 'DOOR MAT NEW ENGLAND' (StockCode: 48187) with total sales of $202.5.
For Italy, the product that generates the most revenue is 'POSTAGE' (StockCode: POST) with total sales of $1524.0.
For Japan, the product that generates the most revenue is 'ROUND SNACK BOXES ,SET 4, FRUITS ' (StockCode: 22328) with total sales of $3794.3999999999996.
For Korea, the product that generates the most revenue is 'TROPICAL  HONEYCOMB PAPER GARLAND ' (StockCode: 21201) with total sales of $100.80000000000001.
For Lebanon, the product that generates the most revenue is 'SET/4 WHITE RETRO STORAGE CUBES ' (StockCode: 84078A) with total sales of $39.95.
For Lithuania, the product that generates the most revenue is 'FELTCRAFT PRINCESS LOLA DOLL' (StockCode: 22750) with total sales of $180.0.
For Malta, the product that generates the most revenue is 'VICTORIAN SEWING KIT' (StockCode: 85178) with total sales of $122.39999999999999.
For Netherlands, the product that generates the most revenue is 'WHITE HANGING HEART T-LIGHT HOLDER' (StockCode: 85123A) with total sales of $9683.9.
For Nigeria, the product that generates the most revenue is 'Adjustment by john on 26/01/2010 16' (StockCode: ADJUST) with total sales of $27.82.
For Norway, the product that generates the most revenue is 'WOODEN ADVENT CALENDAR CREAM' (StockCode: 22946) with total sales of $306.0.
For Poland, the product that generates the most revenue is 'STRAWBERRY CERAMIC TRINKET BOX' (StockCode: 21232) with total sales of $150.0.
For Portugal, the product that generates the most revenue is 'POSTAGE' (StockCode: POST) with total sales of $2347.6.
For RSA, the product that generates the most revenue is 'PINK  SPOTTY PLATE ' (StockCode: 21243) with total sales of $152.72.
For Singapore, the product that generates the most revenue is 'REGENCY CAKESTAND 3 TIER' (StockCode: 22423) with total sales of $751.8.
For Spain, the product that generates the most revenue is 'POSTAGE' (StockCode: POST) with total sales of $2655.0.
For Sweden, the product that generates the most revenue is 'POSTAGE' (StockCode: POST) with total sales of $2059.54.
For Switzerland, the product that generates the most revenue is 'POSTAGE' (StockCode: POST) with total sales of $2685.0.
For Thailand, the product that generates the most revenue is 'SET OF 2 TINS VINTAGE BATHROOM ' (StockCode: 22497) with total sales of $360.0.
For USA, the product that generates the most revenue is 'TOAST ITS - I LOVE YOU ' (StockCode: 21355) with total sales of $452.40000000000003.
For United Arab Emirates, the product that generates the most revenue is 'Manual' (StockCode: M) with total sales of $253.0.
For United Kingdom, the product that generates the most revenue is 'REGENCY CAKESTAND 3 TIER' (StockCode: 22423) with total sales of $143030.66.
For Unspecified, the product that generates the most revenue is 'FAIRY CAKE FLANNEL ASSORTED COLOUR' (StockCode: 21108) with total sales of $206.54999999999998.
For West Indies, the product that generates the most revenue is 'BOX OF 9 PEBBLE CANDLES' (StockCode: 20886) with total sales of $46.8.
```

Questions about Simple Regression Models

- Build a linear regression model that best predicts the total sales per month for this business by using appropriate variables. Compare it to the previous prediction. Calculate the MSE and R2 Score.
- Build a KNN regression model that best predicts the total sales per month for this business by using appropriate variables. Use hyperparameter tuning to determine the best value of K to use in the model. Compare it to the previous predictions and results from the linear regression model

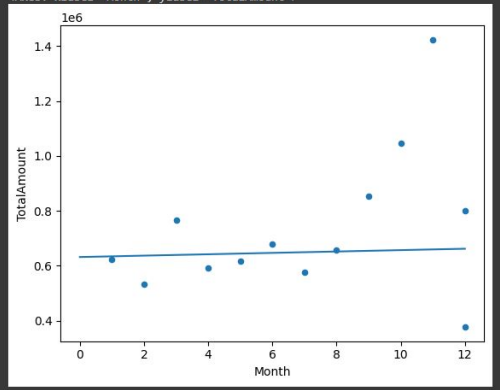
Build a linear regression model that best predicts the total sales per month for this business by using appropriate variables. Compare it to the previous prediction. Calculate the MSE and R2 Score.

```
[ ]  monthly_sales = df.groupby(df['InvoiceDate'].dt.to_period('M'))['TotalAmount'].sum().reset_index()
     monthly_sales['Month'] = monthly_sales['InvoiceDate'].dt.month
     X = monthly_sales[['Month']]
     y = monthly_sales['TotalAmount']

     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

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

     y_pred = model.predict(X_test)

     mse = mean_squared_error(y_test, y_pred)
     r2 = r2_score(y_test, y_pred)

     LMCoef = model.coef_
     LMInter = model.intercept_

     print("Linear Regression Model Parameters - ", "Coefficients: ", LMCoef, " Model Intercept: ", LMInter, "\n")

     print("Mean Squared Error: ", mse)
     print("R^2 Score: ", r2)
```

```
Linear Regression Model Parameters -  Coefficients:  [2513.43628315]  Model Intercept:  631755.3146577062

Mean Squared Error:  213762645808.79013
R^2 Score:  -1.6924138549607326
```

```
X_new = pd.DataFrame()
X_new["Month"] = np.linspace(0, 12, num=60)

y_new_ = pd.Series(
    model.predict(X_new),
    index=X_new["Month"]
)

monthly_sales.plot.scatter(x="Month", y="TotalAmount")
y_new_.plot.line()
```

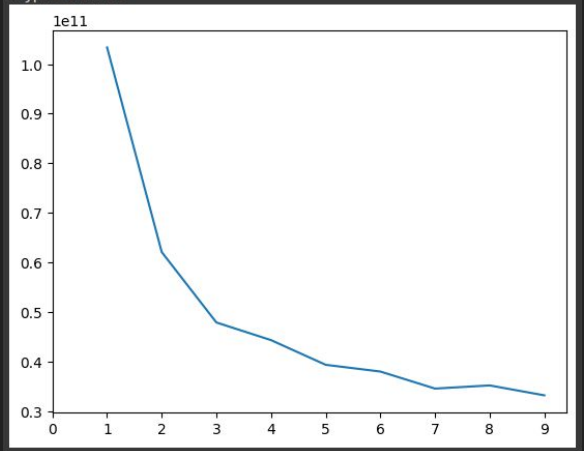
```
<Axes: xlabel='Month', ylabel='TotalAmount'>
```



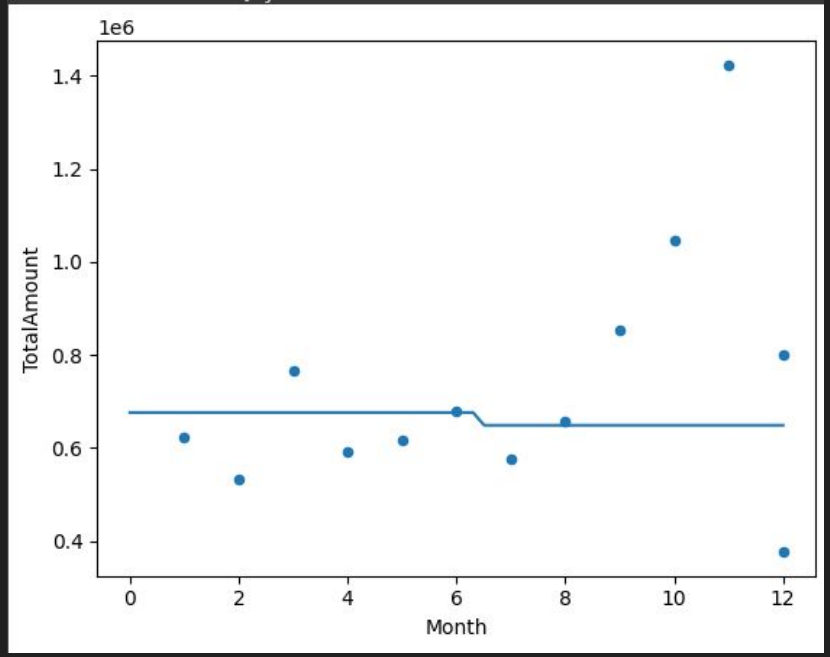
Build a KNN regression model that best predicts the total sales per month for this business by using appropriate variables. Use hyperparameter tuning to determine the best value of K to use in the model. Compare it to the previous predictions and results from the linear regression model

```
monthly_sales = df.groupby(df['InvoiceDate'].dt.to_period('M'))['TotalAmount'].sum().reset_index()
monthly_sales['Month'] = monthly_sales['InvoiceDate'].dt.month
X = monthly_sales[['Month']]
y = monthly_sales['TotalAmount']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Hyperparameter Tuning for KNN Model

def get_cv_error(k):
  knn_model = make_pipeline(
      StandardScaler(),
      KNeighborsRegressor(n_neighbors=k)
  )

  cv_errs = -cross_val_score(knn_model, X=X_train, y=y_train,
                             scoring="neg_mean_squared_error", cv=10)
  return cv_errs.mean()

ks = pd.Series(range(1, 10))
ks.index = range(1, 10)
test_errs = ks.apply(get_cv_error)

test_errs.plot.line(xticks=range(0,10))
test_errs.sort_values()
```

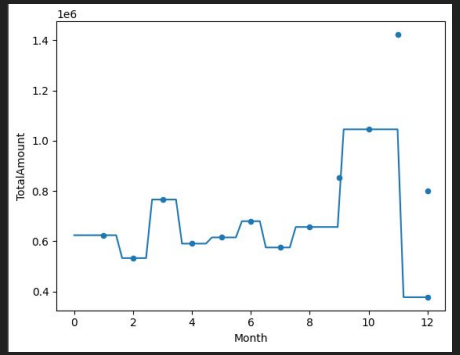
```
9    3.324752e+10
7    3.460870e+10
8    3.524540e+10
6    3.804411e+10
5    3.939829e+10
4    4.437263e+10
3    4.793112e+10
2    6.212064e+10
1    1.033543e+11
dtype: float64
```



```
[ ]  knn_model = KNeighborsRegressor(n_neighbors = 9)
     knn_model.fit(X_train, y_train)

     y_pred = knn_model.predict(X_test)

     knn_mse = mean_squared_error(y_test, y_pred)
     knn_r2 = r2_score(y_test, y_pred)

     print("Mean Squared Error: ", knn_mse)
     print("R^2 Score: ", knn_r2)

     Mean Squared Error:  221201646981.31482
     R^2 Score:  -1.7861106266684068
```




```
# KNN Model with K = 1
knn_model = KNeighborsRegressor(n_neighbors = 1)
knn_model.fit(X_train, y_train)

y_pred = knn_model.predict(X_test)

knn_mse = mean_squared_error(y_test, y_pred)
knn_r2 = r2_score(y_test, y_pred)

X_new = pd.DataFrame()
X_new["Month"] = np.linspace(0, 12, num=60)

# Make predictions at those feature values.
y_new_ = pd.Series(
    knn_model.predict(X_new),
    index=X_new["Month"]
)

# Plot the predictions.
monthly_sales.plot.scatter(x="Month", y="TotalAmount")
y_new_.plot.line()
```



Questions Involving Clustering

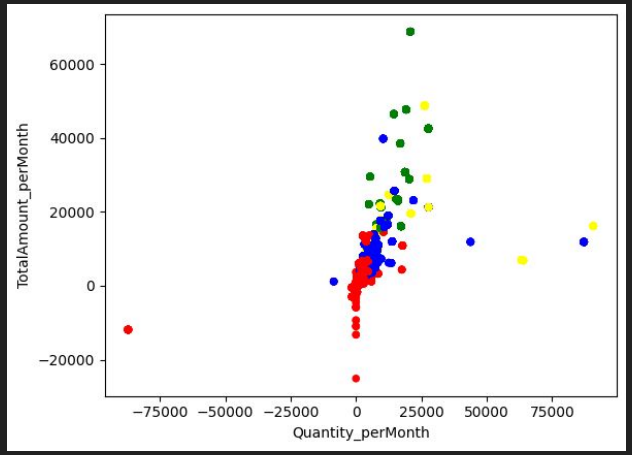
- Can customers be segmented based on purchasing behavior (frequent buyer, seasonal buyer, occasional buyer, etc.) through the use of clustering?
- Show a plot of between quantity a customer purchased for a given month and total amount spent for that month
- Limit plot size to remove extreme observations and allow for break-down of clusters
- Plot quantity a customer purchased for a given year and total amount spent for that year
- Comparing Quantity Per Month to Quantity Per Year
- Comparing Quantity Per Month to Quantity Per Year excluding unusual observations
- Comparing Amount Purchased Per Month to Amount Purchased Per Year
- Comparing Amount Purchased Per Month to Amount Purchased Per Year excluding unusual observations

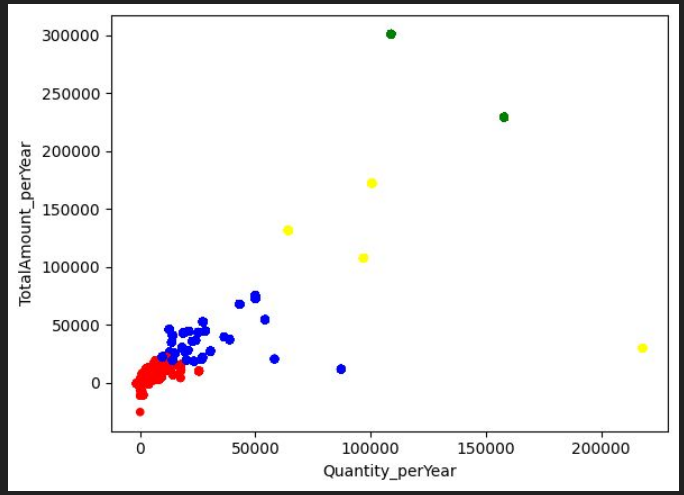
Can customers be segmented based on purchasing behavior (frequent buyer, seasonal buyer, occasional buyer, etc.) through the use of clustering?

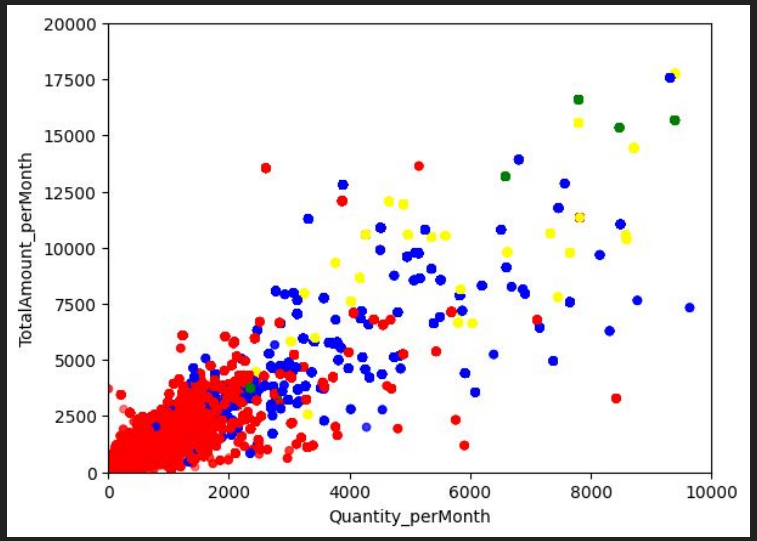
```
[ ]  # Feature Engineering Puchase Variables For Each Unique Customer

     MonthlyPurchaseQuantity = df.groupby(['Customer ID', 'Month'])['Quantity'].sum().reset_index()
     MonthlyPurchaseQuantity
     df = pd.merge(df, MonthlyPurchaseQuantity, on= ['Customer ID', 'Month'], suffixes= ['', '_perMonth'])

     MonthlyPurchaseAmount = df.groupby(['Customer ID', 'Month'])['TotalAmount'].sum().reset_index()
     MonthlyPurchaseAmount
     df = pd.merge(df, MonthlyPurchaseAmount, on= ['Customer ID', 'Month'], suffixes= ['', '_perMonth'])

     YearlyPurchaseQuantity = df.groupby(['Customer ID', 'Year'])['Quantity'].sum().reset_index()
     YearlyPurchaseQuantity
     df  = pd.merge(df, YearlyPurchaseQuantity, on= ['Customer ID', 'Year'], suffixes= ['', '_perYear'])

     YearlyPurchaseAmount = df.groupby(['Customer ID', 'Year'])['TotalAmount'].sum().reset_index()
     YearlyPurchaseAmount
     df = pd.merge(df, YearlyPurchaseAmount, on= ['Customer ID', 'Year'], suffixes= ['', '_perYear'])
```

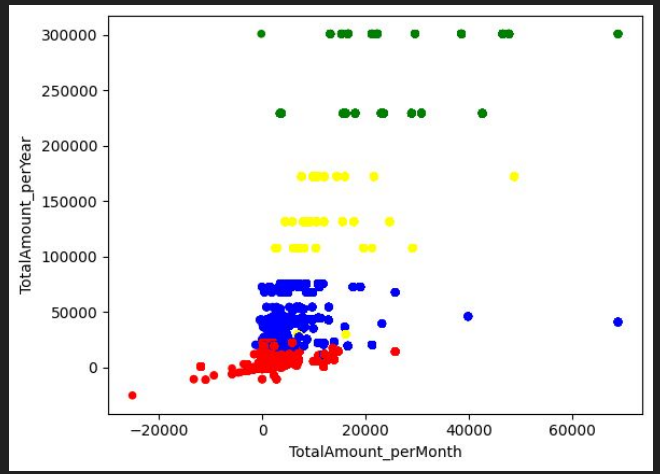
```
# K - Means Cluster
X_train = df[['Quantity_perMonth', 'TotalAmount_perMonth', 'Quantity_perYear',  'TotalAmount_perYear']]

#scaler = StandardScaler()
#scaler.fit(X_train)
#X_train_std = scaler.transform(X_train)

model = KMeans(n_clusters=4) # 4 Clusters were chosen as this seems to provide the most distinct clusters
model.fit(X_train)

purchaseColors = pd.Series(model.labels_).map({
    0: "blue",
    1: "yellow",
    2: "green",
    3: "red"
})

purchaseClusters = model.labels_

df['PurchaseClusters'] = purchaseClusters
```

Clusters

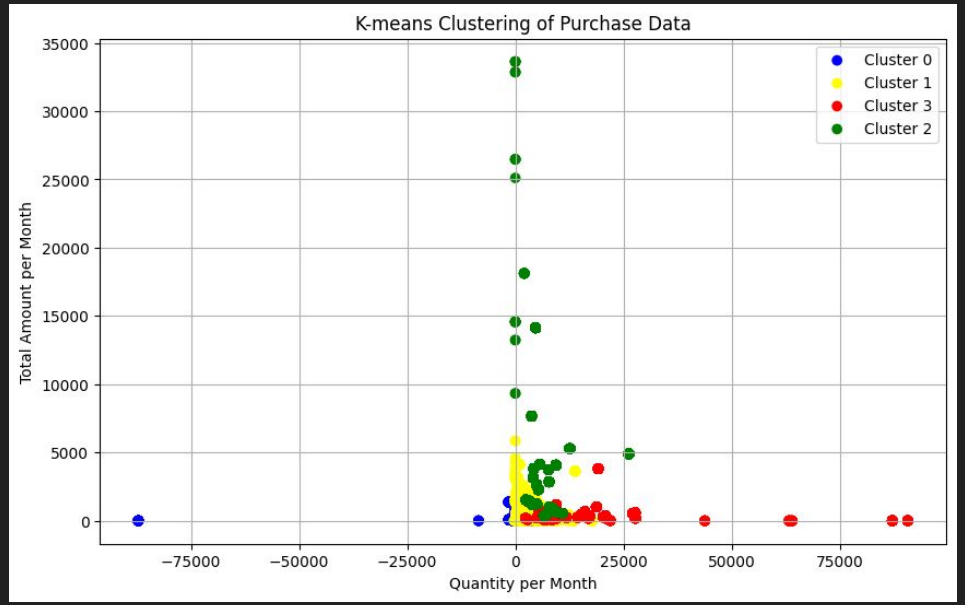

TotalAmount_perMonth
Quantity_perMonth
60000
40000
20000
0
−20000
−75000
−50000
−25000
0
25000
50000
75000


TotalAmount_perYear
Quantity_perYear
300000
250000
200000
150000
100000
50000
0
0
50000
100000
150000
200000


TotalAmount_perMonth
Quantity_perMonth
20000
17500
15000
12500
10000
7500
5000
2500
0
0
2000
4000
6000
8000
10000


TotalAmount_perYear
TotalAmount_perMonth
300000
250000
200000
150000
100000
50000
0
−20000
0
20000
40000
60000

```
plt.figure(figsize=(10, 6))

for cluster_label in df['PurchaseClusters'].unique():
    cluster_data = df[df['PurchaseClusters'] == cluster_label]
    plt.scatter(cluster_data['Quantity_perMonth'], cluster_data['TotalAmount_perMonth'], c=purchaseColors[cluster_data.index], label=f'Cluster {cluster_label}')

plt.title('K-means Clustering of Purchase Data')
plt.xlabel('Quantity per Month')
plt.ylabel('Total Amount per Month')
plt.legend()
plt.grid(True)
plt.show()
```



Questions Involving Classification Models

- Can we determine if a customer will be a repeat customer based on purchase history using classification models? **(**Use the oldest 80% of the sales data in training a classification model, and use the most recent 20% as a validation set for evaluating accuracy of our predictions. Use feature engineering for whether or not there was a repeat customer.)

Can we determine if a customer will be a repeat customer based on purchase history using classification models? (Use the oldest 80% of the sales data in training a classification model, and use the most recent 20% as a validation set for evaluating accuracy of our predictions. Use feature engineering for whether or not there was a repeat customer.)

```
train_size = 0.01  # Use only 1% of the data for training. Using 5% is not finished even after 15 mins.
split_index = int(train_size * len(df))
train_df = df.iloc[:split_index]
test_df = df.iloc[split_index:]

x_predictors = train_df[['StockCode_str', 'CustomerID_str', 'Country', 'Price', 'Month', 'Day', 'Hour', 'Minute']]
y_result = train_df['Return']

ct = make_column_transformer(
    (StandardScaler(), ['Price']),
    (OneHotEncoder(handle_unknown='ignore'), ['StockCode_str', 'CustomerID_str', 'Country']),
    remainder='passthrough'
)

class_model_return = make_pipeline(
    ct,
    KNeighborsClassifier(n_neighbors=5)  # You can experiment with different classifiers here
)

class_model_return.fit(x_predictors, y_result)

x_val = test_df[['StockCode_str', 'CustomerID_str', 'Country', 'Price', 'Month', 'Day', 'Hour', 'Minute']]
y_val = test_df['Return']

accuracy = class_model_return.score(x_val, y_val)
print("Validation Accuracy:", accuracy)
```

```
Validation Accuracy: 0.9316526312478983
```

```
# Define the parameter grid
param_grid = {
    "kneighborsclassifier__n_neighbors": range(1, 10)
}

# Set up grid search
grid_search = GridSearchCV(
    class_model_return,
    param_grid=param_grid,
    scoring="accuracy",
    cv=10
)

# Perform grid search
grid_search.fit(x_train, y_train)

# Get the best parameters and best model
best_params = grid_search.best_params_
best_model = grid_search.best_estimator_

# Calculate accuracy using cross-validation
accuracy = cross_val_score(
    best_model,
    x_train,
    y_train,
    scoring="accuracy",
    cv=10
).mean()

# Print results
print("Best Parameters:", best_params)
print("Accuracy:", accuracy)
```

```
Best Parameters: {'kneighborsclassifier__n_neighbors': 6}
Accuracy: 0.9566572579257169
```

Conclusion