Astoria - Coffee Shop Sales

LINDSAY 2024

The Mission

- Are Sales Consistent?
- What are the Gap in Sales?
- How can I can we fill those gaps ?
- What are the limitations?

Who does this information Help?

The Coffee shop owner

Managers

Locals

Where is the information From?

- How did this transpire?
- Kaggle : https://www.kaggle.com/code/ahmedabbas757/coffee-shop-sales/input
- What does this Data contain?
  - Transactions of 7 months
  - Contained 4 Restaurants.

Let's Wrangle the Data

- Determined relevant columns
- Evaluated the Shape of DF
- What do we see in the columns?
- What are the types of Data we have?

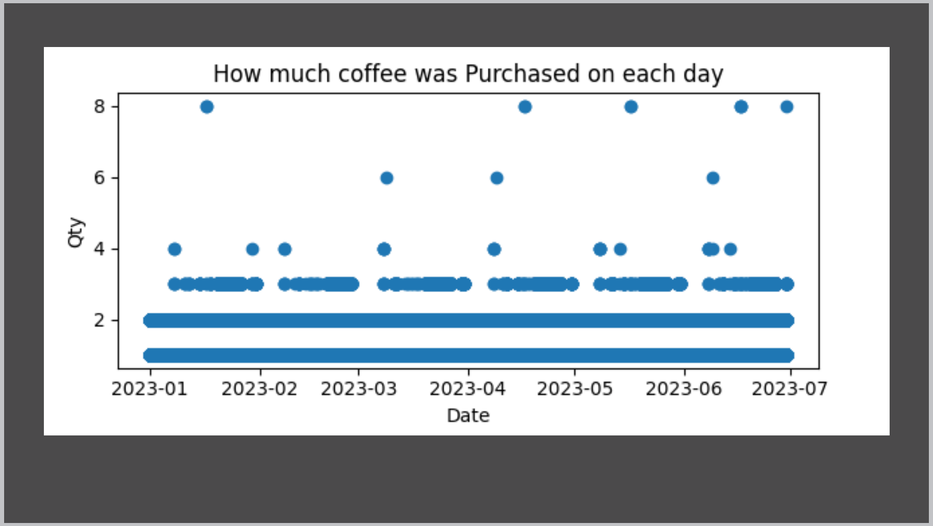
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 149116 entries, 0 to 149115
Data columns (total 11 columns):
 #   Column            Non-Null Count   Dtype         
---  ------            --------------   -----         
 0   transaction_id    149116 non-null  int64         
 1   transaction_date  149116 non-null  datetime64[ns]
 2   transaction_time  149116 non-null  object        
 3   transaction_qty   149116 non-null  int64         
 4   store_id          149116 non-null  int64         
 5   store_location    149116 non-null  object        
 6   product_id        149116 non-null  int64         
 7   unit_price        149116 non-null  float64       
 8   product_category  149116 non-null  object        
 9   product_type      149116 non-null  object        
 10  product_detail    149116 non-null  object        
dtypes: datetime64[ns](1), float64(1), int64(4), object(5)
memory usage: 12.5+ MB
```

```
df.shape
```

```
(149116, 11)
```

	Date	Time	Qty	Store	Price	Product
0	2023-01-01	07:06:11	2	Lower Manhattan	3.00	Coffee
1	2023-01-01	07:08:56	2	Lower Manhattan	3.10	Tea
2	2023-01-01	07:14:04	2	Lower Manhattan	4.50	Drinking Chocolate
3	2023-01-01	07:20:24	1	Lower Manhattan	2.00	Coffee
4	2023-01-01	07:22:41	2	Lower Manhattan	3.10	Tea

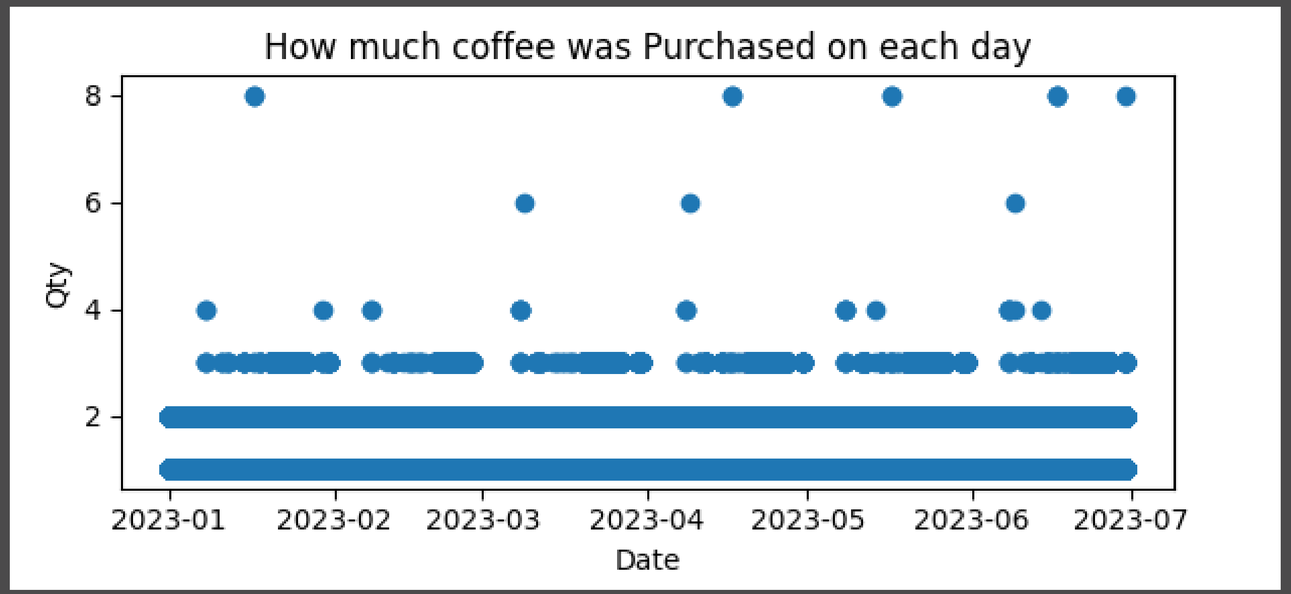


How are we going to Evaluating the Data?

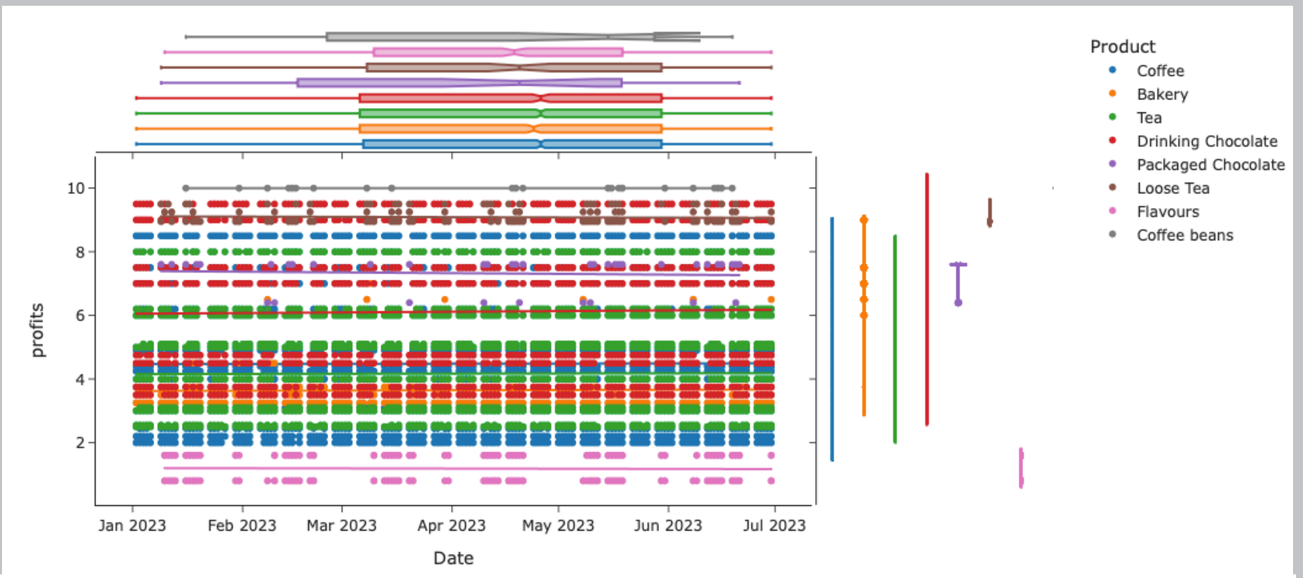
- Multi-Variant Time-Series Data
  - Multiple Variables to track
  - Time-Series
    - 7 months
    - Morning, Afternoon, Evening
    - Weekend VS week day

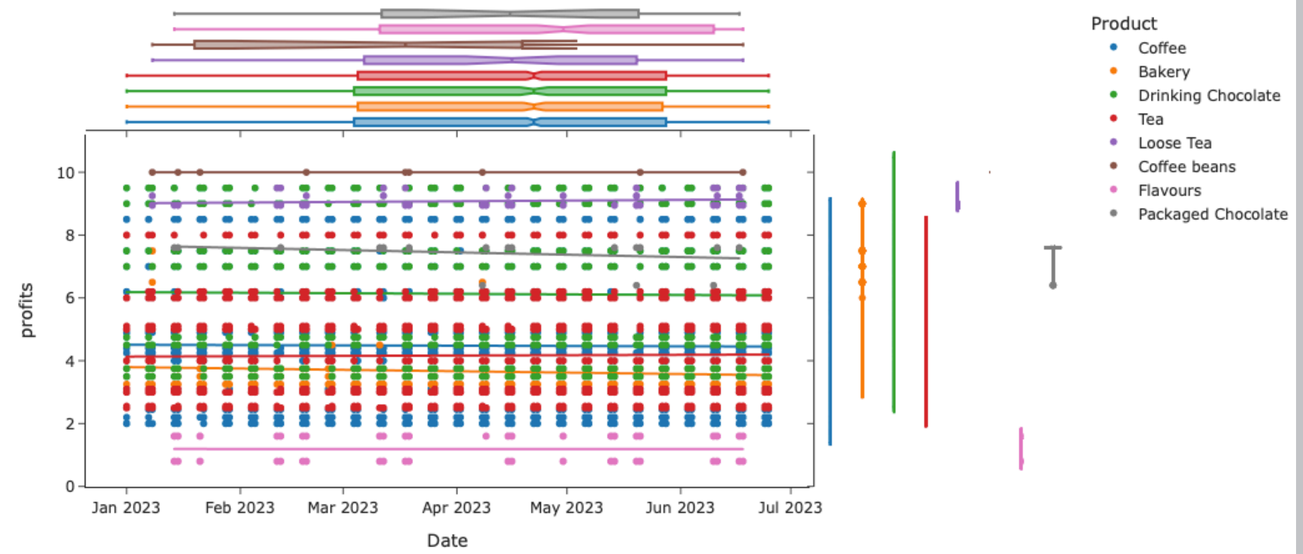
```
#new column to create a distinct period of time morning, day, night
df['Period'] = pd.cut(
    pd.to_datetime(df['Time'], format='%H:%M:%S').dt.hour,
    bins=[0, 6, 12, 18, 24],
    labels=['Night', 'Morning', 'Afternoon', 'Evening'],  # Corrected labels
    right=False  # Use right=False to ensure that 24:00 falls into the last bin
)
```

Quick View of Overall Sales



What is Astoria Selling?



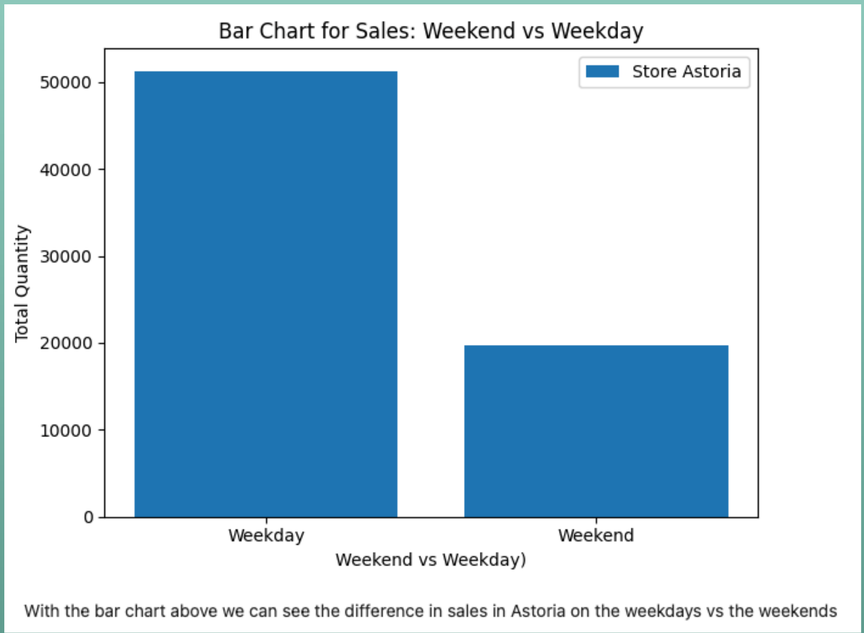


How are the sales?

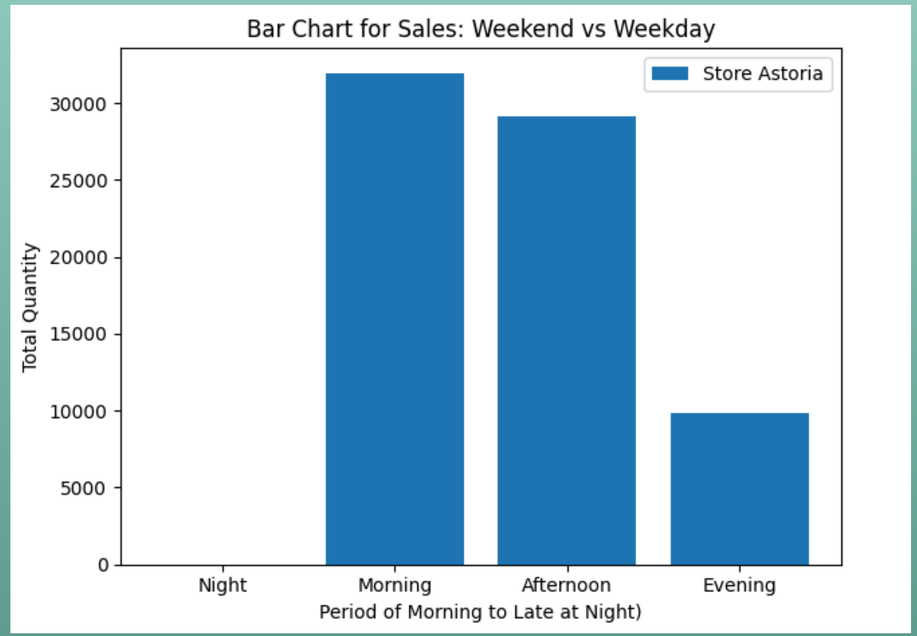
What is the most popular?

- What are the Outliers?
  - Coffee beans
  - Flavours
  - Packaged Chocolates
  - Loose Tea

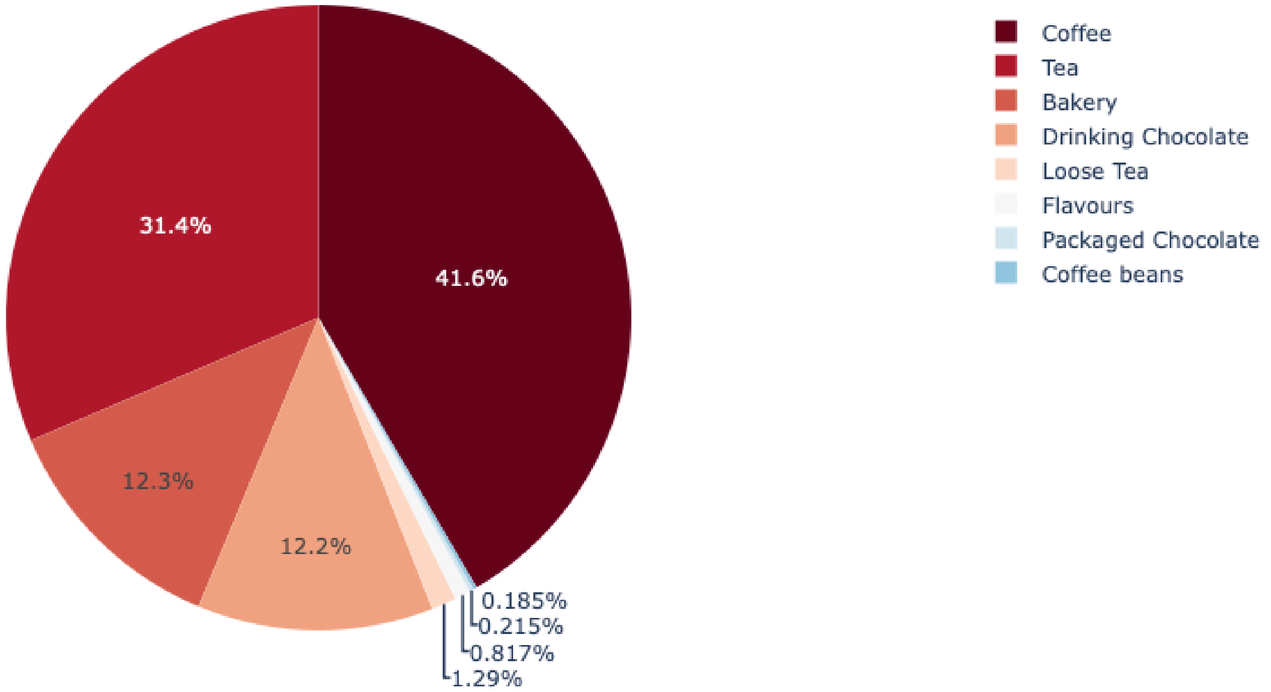
What do we see with the Consistent sales?



With the bar chart above we can see the difference in sales in Astoria on the weekdays vs the weekends



Sales by Product



Modeling / Forecasting the Data

- LSTM
- GRU
- XGBoost
- VAR

Each Model

```
def create_sequences(x, y, seq_length):
    """
    Creates sequences from input data for an LSTM.

    Args:
      x: Input data (features) as a NumPy array.
      y: Target data (labels) as a NumPy array.
      seq_length: The length of each sequence.

    Returns:
      X: A NumPy array containing the input sequences.
      y: A NumPy array containing the corresponding target values.
    """

    X, Y = [], []
    for i in range(len(x) - seq_length):
        X.append(x[i:(i + seq_length)])
        Y.append(y[i + seq_length])
    return np.array(X), np.array(Y)


seq_length = 10
new_x, new_y = create_sequences(X, y, seq_length)

print("X shape:", new_x.shape)
print("Y shape:", new_y.shape)
```

```
X shape: (49744, 30, 17)
Y shape: (49744,)
```

```
train_size = int(len(X) * 0.8)
X_train, X_test = new_x[0:train_size], new_x[train_size:]
Y_train, Y_test = new_y[0:train_size], new_y[train_size:]
```

```
X_train = X_train.astype(np.float32)
Y_train = Y_train.astype(np.float32)

X_test = X_test.astype(np.float32)
Y_test = Y_test.astype(np.float32)
```

LSTM : Long term Short term memory

- Uses sequences
- Form the data into a 3D array

GRU : Gated Recurrent Unit

- Gates to control the flow of dependenc
- Uses sequences

XGBoost: Extreme Gradient Boosting

- Uses Bagging and Boosting
- Capture intricate patterns in time-series data

VAR : Vector Auto Regression

- Max lags weren't optimal

for this times series

```
tscv = TimeSeriesSplit(n_splits=5)

for train_index, test_index in tscv.split(X):
    # Split data into training and testing sets
    X_train_2d, X_test_2d = X.iloc[train_index], X.iloc[test_index]
    y_train_2d, y_test_2d = y.iloc[train_index], y.iloc[test_index]

    # Ensure y is reshaped as a 1D array
    y_train_1d = y_train_2d.values.ravel()
    y_test_1d = y_test_2d.values.ravel()

    # Scale inputs
    scalerX = StandardScaler()
    X_train_scaled = scalerX.fit_transform(X_train_2d)
    X_test_scaled = scalerX.transform(X_test_2d)

    print("X_train_scaled shape:", X_train_scaled.shape)
    print("y_train_1d shape:", y_train_1d.shape)

    # Train the XGBoost model
    reg = xgboost.XGBRegressor(objective='reg:squarederror', n_estimators=1000, nthread=24)
    reg.fit(X_train_scaled, y_train_1d)

    # Predict the test data
    predictions_xgb = reg.predict(X_test_scaled)

    # Calculate RMSE
    rmse_xgb = sqrt(mean_squared_error(y_test_1d, predictions_xgb))
    print("XGBoost - Root Mean Square Error (RMSE): %.3f" % rmse_xgb)

    # Visualize predictions vs. actual
    import matplotlib.pyplot as plt
    plt.plot(y_test_1d, label='Actual')
    plt.plot(predictions_xgb, label='Predicted', linestyle='--')
    plt.legend()
    plt.title('XGBoost Predictions vs Actual')
    plt.show()
```

WINNER : XGBOOST

```
#Comparing all models

print("LSTM Root Mean Squared Error: ", best_rmse)
print("GRU Root Mean Squared Error: ", rmse_gru)
print("XGBoost Root Mean Squared Error: ", rmse_xgb)
```

```
LSTM Root Mean Squared Error:  0.4854950414535987
GRU Root Mean Squared Error:  0.4849633519838093
XGBoost Root Mean Squared Error:  1.0946075704543654e-05
```

Recommendations

- Taste testing events for the evening
  - Inspire to purchase Shelf items
  - Taste test coffee and tea
    - Helps with hearing customers ideas
- Loyalty Program
- Opening a hour earlier
  - New York City


Thank you!