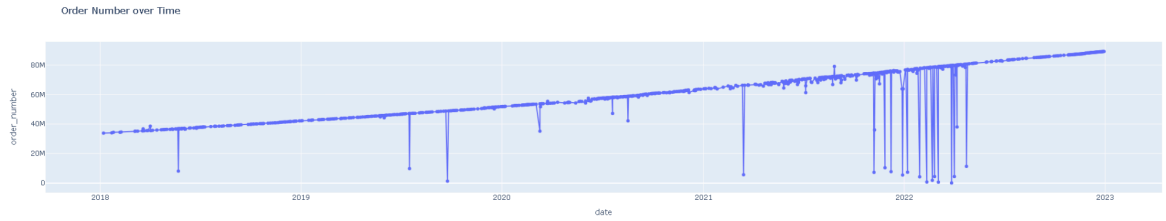
Predicting the revenue of ABC-Retail

First Data

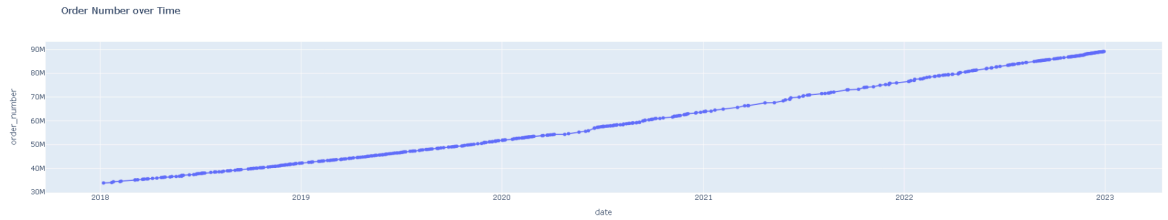
Initial data provided contained three sheets namely ‘order_numbers’, ‘transaction_data’ and ‘reported_data’. On first look, the ‘transaction_data’ contained 2 columns. These were ‘date’ and ‘order_number’. The shape of the data is (856,2). The dates included were not continuous with dates from each day of the month which suggested that it was either missing this data or there weren't any transactions on all days of each month. The date started from 7th of January 2018 till 30th of December 2022. Instructions provided regarding the data explained that the order numbers were supposed to increase according to date and should not decrease. But as there were bad data points, the data required to be cleaned.I loaded the data onto a pandas dataframe and created a new column called ‘order_number_difference’ which was a difference between the order-numbers of two subsequent dates. Some of the differences showed a negative value, which gave me the impression that the first of the two subsequent pairs was a bad datapoint for whenever the difference was a negative value. My first thought was to remove these bad data points to clean the data. Then I noticed that some of the dates were repeated and these dates had multiple ‘order_numbers’ accordingly. I confirmed this theory by adding another column called ‘date_difference’ which was the difference of two dates on which I saw multiple values of zero. This did throw me into a loop as I noticed that by removing some of the bad data points I might be creating newer bad data points and that I might end up losing most of the data in the given data set. I checked to see the outliers by creating an interactive plot using plotly as seen in the image below.



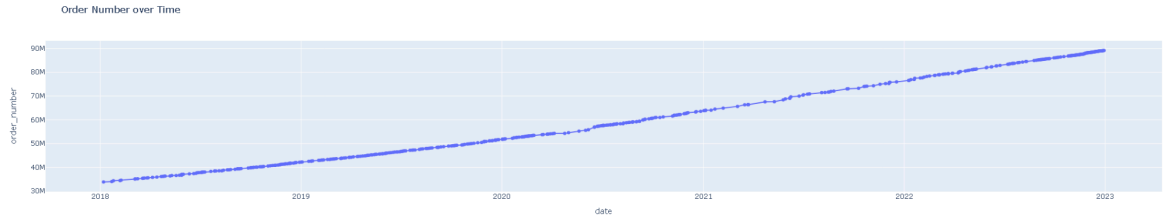
I created a new column called ‘data_integrity’ and filled the values with ‘True’ initially. Then I iterated through the indices of the ‘order_number’ column to check the condition where previous_order<current_order<next_order where each of these were 3 subsequent ‘order_number’. If this was not the case, then the ‘data integrity’ of the ‘current_order’ was set to ‘False’.

Now I proceeded to only put the rows with ‘data_integrity’ set to ‘True’ onto a new data frame called ‘df_good’. I also created another dataframe called ‘df_good2’ with only ‘data_integrity set to ‘True' and with the max value of the order_number for the rows with the same subsequent date. This was to experiment with how the model performs with both approaches to the problem.

The interactive plot for the ‘df_good’ is given below,



The interactive plot for the ‘df_good2’ is given below,



Both approaches show a very similar curve and no more outliers are visible to the naked eye or upon closer inspection. Both have a shape of (498,5).

Second Data

The second data sheet called ‘transaction_data’ contained ‘date’, ‘total_spend_index and ‘weekly_active_users_index’. The dates started from the first of January 2018 till the 31st of December 2022. Now this made me wonder as to how I might be able to merge the two data sets from the two sheets as I needed them to be able to predict the data for the third sheet. The data didn't look like it needed any cleaning as the subsequent dates did not have any missing values.

I created a new dataframe called ‘df_spend’ using the data from the sheet called ‘transaction_data’. I checked for any correlation between the data in ‘total_spend_index’ and ‘weekly_active_users_index’ and the value was ‘0.7163052468998714’. As the feature ‘weekly_active_users_index’ is highly correlated to ‘total_spend_index’, I decided it wasn't worth including this feature for training.

Processing

The third sheet called ‘reported_data’ contains three columns named ‘period’, ‘start_date’, ‘end_date’ and ‘revenue_index’ for different quarters for the years from 2018 to the end of 2022. The duration for these quarters do not conform to the usual 4 months in a quarter that you know about.To process the data further for the next part of the task, I put this data on a dataframe called ‘df_period’. By using ‘df_good2’ (as both df_good and ‘df_good2’ both had similar values I opted for using ‘df_good2’ as it had the additional conditioning that was performed on it in comparison with ‘df_good’). I re-indexed the dates according to the

‘start_date’ and ‘end_date’ for each quarter accordingly for ‘df_good2’ and interpolated the ‘order_number’. I then found the difference between the orders so as to estimate the values of unknown data points that fall in between these dates. The differences were then summed up to find the ‘total_orders_for_period’.

So as to handle the ‘spend_index’, the ‘spend index’ was filtered for the dates between the ‘start-date’ and ‘end_date’ (including those particular dates). These were then summed up into ‘total_spend_for_period’.

‘df_period’ was then stored into ‘data’ to be used for the model.

Models

I created two models, namely ‘Model_1’ and ‘Model_2’ to see what features were the best at predicting the revenue index for Q4 of 2022

‘Model_1’ contains 4 features,

1. ‘total_days’ which is the number of days between ‘start_date’ and ‘end_date’
2. ‘start_year’ which is the year in which the ‘start_date’ begins
3. ‘total_orders’
4. ‘total_spend_index’

‘Model_2’ contains 4 features,

1. ‘total_days’ which is the number of days between ‘start_date’ and ‘end_date’
2. ‘start_year’ which is the year in which the ‘start_date’ begins
3. ‘total_spend_index’

Both models had the target of ‘revenue_index’. The data was named as X for features and y for target. The data was split into ‘X_train’, ‘X_test’, ‘y_train’, ‘y_test’ with a ‘test_size’ of ‘0.2’ and ‘random_state’ of ‘42’. Different models were initialised for training and these were ‘Lasso’, 'Ridge’, ‘Ridge with cross-validation’, ‘Gradient Boost’ and ‘Xgboost’. Refer to the image below for reference. These models were trained using ‘X_train’ and ‘y_train’ and predictions were generated using ‘X_test’.

```
# Initialising the models
lasso_model = Lasso(alpha=0.1)
ridge_model = Ridge(alpha=0.1)
ridge_cv_model = RidgeCV(alphas=[0.1, 0.25, 0.5, 1.0, 10.0], cv=5)  # Ridge with Cross-Validation
gboost_model = GradientBoostingRegressor(n_estimators=100, learning_rate=0.1, max_depth=5)
xgb_model = xgb.XGBRegressor(objective='reg:squarederror', n_estimators=100, learning_rate=0.1, max_depth=5)


# Training the models
lasso_model.fit(X_train, y_train)
ridge_model.fit(X_train, y_train)
ridge_cv_model.fit(X_train, y_train)
gboost_model.fit(X_train, y_train)
xgb_model.fit(X_train, y_train)

# Generating predictions for the test set
lasso_preds = lasso_model.predict(X_test)
ridge_preds = ridge_model.predict(X_test)
ridge_cv_preds = ridge_cv_model.predict(X_test)
gboost_preds = gboost_model.predict(X_test)
xgb_preds = xgb_model.predict(X_test)
```

The weights for the models were adjusted as shown below,

```
# Combining predictions using different weights
weights = {
    'lasso': 0.15,
    'ridge': 0.1,
    'ridge_cv': 0.15,
    'gboost': 0.1,
    'xgb':0.5
}
```

The final prediction is a weighted average of the individual model predictions and loaded onto ‘final_preds’. The ‘mean_squared_error’ is calculated using ‘y_test’ and ‘final_preds’ to evaluate the ensemble model. I then created a new function to make predictions using the ensemble model.

Since we don't have total_orders and total_spend_index at test time, we'll have to use only the features derived from dates (‘total_days’, ‘start_year’).

For ‘model_1’ we will load the following for the ‘input_data’,

1. ‘total_days’ which contains the total days between ‘start_date’ and ‘end_date’.
2. ‘start_year’ which is the year in which the ‘start_date’ begins.
3. ‘total_order’ has zero as a placeholder value.
4. ‘total_spend_index’ has zero as a placeholder value.

For ‘model_1’ we will load the following for the ‘input_data’,

1. ‘total_days’ which contains the total days between ‘start_date’ and ‘end_date’.
2. ‘start_year’ which is the year in which the ‘start_date’ begins.
3. ‘total_spend_index’ has zero as a placeholder value.

I then proceeded to make predictions from the individual models which were again ‘Lasso’, 'Ridge’, ‘Ridge with cross-validation’, ‘Gradient Boost’ and ‘Xgboost’. The predictions were then combined using the same weights as before onto ‘final_pred’.

The predicted revenue index for the dates between ‘01/01/2022’ and ‘31/12/2022’ were as follows,

Model_1: 509.17351338549753
Model_2: 507.02415643573954

As Model_1 was the closest to the actual value, I chose Model_1 as the best model with the best features.