A1 IML Report

Urvashi Balasubramaniam and Kabir Vohra
Retail Task Report

Introduction

1. **Introduction** – Briefly describe the dataset and problem.

Dataset: Retail database, used to predict **avg_purchase_value** of the customer

Database size information: 80k rows, 77 features

Original columns: Describing the transaction, previous purchases, time, product, quantity, pricing, location, rating, reviews, role of promotions (both discounts and social media advertisement) as well as engagement with the app and website in detail.

Index(['age', 'gender', 'income_bracket', 'loyalty_program',
'membership_years', 'churned', 'marital_status', 'number_of_children',
'education_level', 'occupation', 'transaction_id', 'transaction_date',
'product_id', 'product_category', 'quantity', 'unit_price',
'discount_applied', 'payment_method', 'store_location',
'transaction_hour', 'day_of_week', 'week_of_year', 'month_of_year',
'avg_purchase_value', 'purchase_frequency', 'last_purchase_date',
'avg_discount_used', 'preferred_store', 'online_purchases',
'in_store_purchases', 'avg_items_per_transaction',
'avg_transaction_value', 'total_returned_items', 'total_returned_value',
'total_sales', 'total_transactions', 'total_items_purchased',
'total_discounts_received', 'avg_spent_per_category',
'max_single_purchase_value', 'min_single_purchase_value',
'product_name', 'product_brand', 'product_rating',
'product_review_count', 'product_stock', 'product_return_rate',
'product_size', 'product_weight', 'product_color', 'product_material',
'product_manufacture_date', 'product_expiry_date', 'product_shelf_life',
'promotion_id', 'promotion_type', 'promotion_start_date',
'promotion_end_date', 'promotion_effectiveness', 'promotion_channel',
'promotion_target_audience', 'customer_zip_code', 'customer_city',
'customer_state', 'store_zip_code', 'store_city', 'store_state',
'distance_to_store', 'holiday_season', 'season', 'weekend',
'customer_support_calls', 'email_subscriptions', 'app_usage',
'website_visits', 'social_media_engagement',
'days_since_last_purchase'],

dtype='object')

Correlation between features: low, as indicated in the heatmap of the jupyter notebook. This means the features are relatively independent, which is beneficial and reduces confusion of the linear regression model.

Datatypes:
Numerical variables (floating point and integer)
Categorical variables (all have less than 10 distinct values, making it ideal for one-hot encoding)

Methodology

- The workflow began with **data viewing and cleaning** using RowZero, pandas (Jupyter Notebook), and terminal tools such as `awk`.
- Missing values were checked, and outlier detection was explored.
- For feature engineering, **categorical and numerical variables** were separated using pandas `select_dtypes`.
- Categorical variables were one-hot encoded, initially through scikit-learn's `OneHotEncoder` combined with a `ColumnTransformer`. However, due to kernel crashes caused by extremely high-cardinality features (such as IDs and date-time stamps with >700k unique values), the process was optimized.
- All encoded columns were converted from boolean → integer (0/1).
- The pipeline was simplified by adopting **`pd.get_dummies()`** for one-hot encoding.
- To handle date and time fields, unique identifiers were discarded or transformed into more meaningful features (e.g., deriving `promotion_time_period` from promotion start and end dates, extracting time-of-day or month indicators).
- Features with no predictive value, such as unique IDs and redundant date fields, were removed.

For correlation analysis:

- **Numerical–numerical correlation** was performed with `df.corr()`.
- **Categorical–numerical correlation** was assessed via **point biserial correlation**.
- **Categorical–categorical dependence** was tested with **chi-square**, though this was abandoned since the target was continuous.

Normalisation

- After cleaning, **MinMax scaling** was applied to normalize the numerical features for model building.

Data filtering

- **`transaction_date`** – redundant since seasonality/week/month/day/time features already exist. → **Dropped**
- **`last_purchase_date`** – could be transformed to "days since last purchase," but limited utility. → **Dropped**
- **`product_manufacture_date`** & **`product_expiry_date`** – already summarized as `product_shelf_life`. → **Dropped**
- **`promotion_start_date`** & **`promotion_end_date`** – engineered into a new feature:
    - **`promotion_time_period`** = number of days between start and end → **Kept**
    - Raw columns → **Dropped**

Correlation Analysis

Using Pearson's correlation for numerical values (`df.corr()`):

- **Target feature**: `avg_purchase_value`
- Overall, **correlations were very low** across numerical features, suggesting weak linear relationships.
- Top correlated numerical features included:
    - `total_returned_items` (+0.0025)
    - `in_store_purchases` (+0.0024)
    - `quantity` (+0.0023)

After one-hot encoding, categorical correlations with the target were tested using **Point Biserial correlation** (`scipy.stats.pointbiserialr`).

- Statistically significant ($p < 0.05$) features included:
    - `gender_Male` (+0.0023)
    - `customer_city_City C` (+0.0025)

- `payment_method_Cash` (+0.0020)
- `loyalty_program_No` (+0.0016)

When combining numerical and categorical features, the top 10 correlations with `avg_purchase_value` (in absolute terms) were:

1. `avg_purchase_value` (self, 1.000)
2. `customer_city_City C` (+0.0026)
3. `total_returned_items` (+0.0025)
4. `in_store_purchases` (+0.0025)
5. `gender_Male` (+0.0023)
6. `quantity` (+0.0023)
7. `payment_method_Cash` (+0.0021)
8. `transaction_id` (+0.0017)
9. `total_items_purchased` (+0.0017)
10. `loyalty_program_No` (+0.0017)

These values confirm ***extremely weak correlations overall***, suggesting more complex (non-linear) relationships could be possible.

Visualisations

Correlation Heatmap

- Heatmap of numerical features showed most cells in **white/grey (almost no correlation)**.
- This confirms **low multicollinearity** – features are mostly independent and unlikely to confuse models.

Scatter & Box Plots

Scatter plots made between numerical features and the target (`avg_purchase_value`).

- **Findings**: No strong linear trends were visible. Continuous-discrete variable mismatches occurred.
- ***Insight***: Relationships may be **non-linear or interaction-based** rather than simple linear associations.

Training

Multiple subsets were created for model training:

- **All training features** (`all_training_data.csv`) – wide feature set.
- **Top correlated subset** (`training_subset.csv`) – using the most correlated categorical + numerical variables.
- **Bottom weakest correlations excluded** (`training_data_minus30.csv`) – testing robustness by removing noise features.
- **Quadratic model** – considering it might be a non-linear model, tried quadratic modelling as well. Did not get robust results.

Project Log (Experimentation Section)

1:40pm
Data viewing and cleaning with RowZero, pandas (jupyter notebook) and terminal awk

3:42 pm
Checking for missing values

3:00pm
Understood interquartile, visualising methods for outlier detection. Settled on z score outlier detection, done separately for numerical and categorical values. Wrote code that selects numerical and categorical data, and then performs correlation testing

4:10pm
Learnt how to do one hot encoding, implementing using scikitlearn

4:20pm
Checking for high and low correlation features with the target feature.
df.corr() works for numerical values only, so doing one-hot encoding before using it.

Need to perform one-hot encoding on the following categorical variables. Of the 77 columns:
(1) Identifying those that have Strings as fields.

-> Since the data type is consistent in a column, check in O(n) by checking all features
-> Figured out pandas code to select dtypes.
numericals = df.select_dtypes(include=["number"]).columns
categoricals = df.select_dtypes(exclude=["number"]).columns

(2) Coming up with a standard rule to perform one-hot encoding (without my manual intervention required)

Initial idea was to count distinct values in every column using:

print(f"The number of distinct product IDs is: {unique_products}")

Then make that unique_products many sub-features and assign binary values.

Can ***instead use inbuilt OneHotEncoder()*** in scikitlearn (as recommended by TF)

5:32pm

```
import sklearn.preprocessing as MinMaxScaler
from sklearn.compose import ColumnTransformer

# correlation analysis -- every variable with target
# Separating numerical and categorical (non-numerical/string) data value columns
numericalCols = df.select_dtypes(include=["number"]).columns.tolist()
categoryCols = df.select_dtypes(exclude=["number"]).columns.tolist()

# One hot encoding using scikitlearn
ct = ColumnTransformer( transformers=[
     ("cat", MinMaxScaler.OneHotEncoder(handle_unknown='ignore'), categoryCols),
     ("num", "passthrough", numericalCols) # passthrough = continue
  ])
"""
Transformation:
cat (transform name) -> apply OneHotEncoder to categoryCols
num (transform name)-> just pass through without applying anything
Called later using ct.fit_transform(df) which outputs the numpy array (without headers)
"""

# Call the above transform (outputs numpy array)
tempEncoded = ct.fit_transform(df)

# Add back the headers and numerical data
cat_features = ct.named_transformers_["cat"].get_feature_names_out(categoryCols)
```

```
all_features = list(cat_features) + numericalCols


# Making it a dataframe again
df_encoded = pd.DataFrame(tempEncoded.toarray(), columns=all_features)
```

Ran it in jupyter notebook – kernel crashed three times (taking upwards of 40 seconds, timing out)

Making the operation lighter

Changed import statements (importing classes separately)

```
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
```

Removing toarray() from the tempEncoded file – as it is already a matrix by default, converting it to Numpy is redundant and occupies too much memory. Instead adding "sparse_output=false" so that it stores the data densely (closer to the numpy array I need, which I'll convert to a dataframe)

The kernel crashed again.

One-hot-encoding is failing because there are several columns with 80k unique values (like IDs), that have no bearing on the model but are being encoded.

Fixes:
-> Check how many distinct values exist per column
-> If all values distinct => delete column (IDs don't inform the model)
-> One hot encode them under a certain threshold

Anticipated future problems
-> replace '0's in the dataset

Cleaning up the data, reducing uniqueness and extracting relevant information:
-> Remove all alphanumeric IDs, if any
-> Transaction_date, last_purchase_date, product_manufacture_date, product_expiry_date, promotion_start_date, promotion_end_date includes exact time and alphanumeric unique information. Extract month and year from this, and a morning/afternoon/evening/night categoriser

Reasons to drop a feature:
If it has low correlation, drop the variable
If it has high number of unique data values

Problem: corr() works on numerical data, and I'd need ANOVA to do it on text. Not ideal.

11:10pm

Retry with sparse matrix (sparse_output=True) and create a sparse numpy array -> but then I'll have to specify many operations that aren't supported myself

11:20pm

Try another encoding method? Binary or target encoding would take up a lot less memory.

Looking at the data again for completely unique values. Many transaction IDs (with 80k unique values) are unique, but they're numerical and thus not being one-hot encoded. They may be removed, however, as their ID number is arbitrary and won't contribute to the correlation

Cleaning up the data, reducing uniqueness and extracting relevant information:

-> Remove all alphanumeric IDs, if any. Found none.

-> Transaction_date, last_purchase_date, product_manufacture_date, product_expiry_date, promotion_start_date, promotion_end_**date includes exact time and alphanumeric completely unique information**. Extract month and year from this, and a morning/afternoon/evening/night categoriser.

12:00am

Handling date and time differently

Testing for correlation, top numerical variables that correlate are the following:

avg_purchase_value 1.000000 (the target itself)
total_returned_items 0.002510
in_store_purchases 0.002482
quantity 0.002272
transaction_id 0.001726
total_items_purchased 0.001661
website_visits 0.001527
customer_zip_code 0.001292
days_since_last_purchase 0.000979
product_review_count 0.000955

10:50am

To test correlation for categorical variables with target feature – tried using ***chi-squared test***

```
from scipy.stats import chi2_contingency
```

```
categoryCols = df.select_dtypes(exclude=["number"]).columns.tolist()

# loop
for i in range(len(categoryCols)):
  for j in range(i + 1, len(categoryCols)):
    col1 = categoryCols[i]
    col2 = categoryCols[j]

    # contingency table keeps one category on x axis, other on y (like a punnett square)
    # summarises the frequency counts for each combo of categories for the 2 vars
    contingency_table = pd.crosstab(df[col1], df[col2])

    # calculating chi-squared statistic
    chi2, p_value, dof, expected = chi2_contingency(contingency_table)

    results[f"{col1} vs {col2}"] = {
      "chi2_statistic": chi2,
      "p_value": p_value,
      "degrees_of_freedom": dof
    }
```

Realised chi square test only works on categorical-categorical comparisons and target variable is numerical and continuous.

12:05pm

Explored point biserial correlation method – but this only works for binary categorical variables.

https://datascientest.com/en/calculate-correlation-between-two-variables-how-do-you-measure-dependence. Instead using ANOVA, as there are several columns with n>2 unique values.

Alternatively, one-hot encode the categorical data and then test correlation using Point Biserial Correlation!

Find number of unique values per column, looping through categorical columns

As predicted, the dates are the only columns with significantly unique values, mostly because they are highly specific with datetimes

-> One hot encode the rest of the variables, as the number of new columns created is less than 10 for all non-date features

OUTPUT
Column name : number of unique values

'gender' : 3
'income_bracket' : 3
'loyalty_program' : 2
'churned' : 2
'marital_status' : 3
'education_level' : 4
'occupation' : 4
'transaction_date' : 795027
'product_category' : 5
'payment_method' : 4
'store_location' : 4
'day_of_week' : 7
'purchase_frequency' : 4
'last_purchase_date' : 789895
'preferred_store' : 4
'product_name' : 4
'product_brand' : 3
'product_size' : 3
'product_color' : 5
'product_material' : 4
'product_manufacture_date' : 794914
'product_expiry_date' : 794875
'promotion_type' : 3
'promotion_start_date' : 789928
'promotion_end_date' : 789906
'promotion_effectiveness' : 3
'promotion_channel' : 3
'promotion_target_audience' : 2
'customer_city' : 4
'customer_state' : 3
'store_city' : 4
'store_state' : 3
'holiday_season' : 2
'season' : 4
'weekend' : 2
'email_subscriptions' : 2
'app_usage' : 3
'social_media_engagement' : 3

12:23pm

-> Extract date info using this method, so we don't have to delete the features– ***https://medium.com/@paghadalsneh/handling-date-and-time-data-in-machine-learning-a-comprehensive-guide-5d30141cbfec***

Analysis: What data could I potentially extract from the date columns?

'transaction_date' : 795027

- Seasonality (extract quarter)
- Week of the year – already a separate feature in my dataset
- Month of the year – already given in "month of year"
- Time of day – already given in "transaction hour"
- Weekend indicator (yes/no binary value) – already done in "day of the week"

Therefore, ***delete this column*** as it provides no new information.

'last_purchase_date' : 789895

- We already have purchase frequency
- Could convert to "days since last purchase" – but this would have to be dynamically updated according to the current date, and still doesn't provide much new information.

Therefore, ***delete this column*** as it provides little new information.

'product_manufacture_date' : 794914
'product_expiry_date' : 794875

- This interval is already represented clearly in **product_shelf_life**

Therefore, ***delete this column*** as it provides no new information.

'promotion_start_date' : 789928
'promotion_end_date' : 789906

- Convert this column to an interval (**promotion_time_period**) that calculates the number of days between start and end date

12:37pm
-> **Verifying promotion_time_period** by manually calculating three rows' intervals – it's correct!
-> Deleting ***promotion_start_date, promotion_end_date***

3:00pm
My previous one-hot encoding code was too complicated. Just using pd.get_dummies() instead.

3:10pm

It outputs True's and False's, so I'll convert this to 1's and 0's for binary values of all columns.

3:36pm

Point biserial testing on categorical data to get relevant features with respect to the target feature. It's a symmetric test so ordering doesn't particularly matter.

Formula: r = (P(1) — P(0)) * sqrt(n(1) * n(0)) / sqrt(P(1) * (1-P(1)) * n(1) + P(0) * (1-P(0)) * n(0))

Using scipy stats to implement this formula.

```
The point biserial correlation coefficient ranges from -1 to 1,
with positive values indicating a positive correlation
and negative values indicating a negative correlation.
Values close to 0 indicate little or no correlation.
```

Insights from https://koshurai.medium.com/point-biserial-correlation-32440e373718

4:17pm

Started correlation testing

5:30pm

Finished correlation testing! Got four categorical variables with high correlation and statistically.

Also found out corr() now works on one-hot encoded data. I have tested the same thing twice, unfortunately, but this got me useful data on the categorical and numerical features separately at least.

IMPORTANT FEATURES (highly correlated variables):

1. avg_purchase_value 1.000000
2. customer_city_City C 0.002578
3. total_returned_items 0.002510
4. in_store_purchases 0.002482
5. gender_Male 0.002337
6. quantity 0.002272
7. payment_method_Cash 0.002060
8. transaction_id 0.001726
9. total_items_purchased 0.001661
10. loyalty_program_No 0.001652

Can now build a model with just these features!

5:57pm

Standardization (MinMaxScaling): Scales features to have a mean of 0 and a standard deviation of 1. Follows formula:

$$X\_normalized = \frac{X - X\_min}{X\_max - X\_min}$$

Implementing this in pandas^

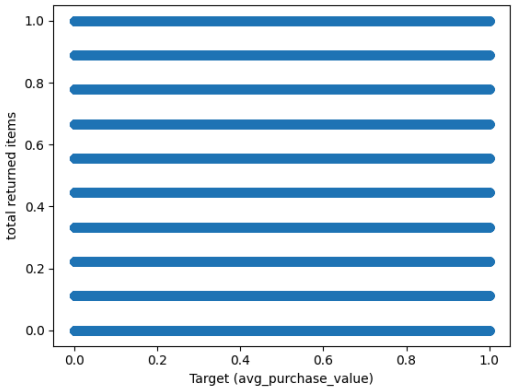
6:16pm

Read about whether post normalisation, one-hot encoded features would contribute more than normalised numerical features (a 1 for example might contribute more than a 0.445552 which is a normalised numerical feature). Learnt this is not the case. Proceeding with visualisations and making the model.

6:21pm

Visualisations

1. Plotted target against total_returned_items (highest correlation numerical feature), but it looked odd.



This I learnt is because the total_returned_items data is discrete while my target feature is basically continuous, thus creating stripes.

7:08pm

2. Coded and interpreted heatmap with non-target features to understand correlation between features. Found that multicollinearity is low as almost all cells not along the variable diagonal are white/grey! This is good for regression.

7:16pm

3. Trying scatter plots for continuous-continuous features (both target and other feature)

8:13pm
-> ***Model building done*** (same as previous models)

9:52pm
-> Training data decisionmaking:

2. Extracted the column names from the dataframe
3. Created new dataframes with all selected columns, ran five training tests (see Training Experimentation for more details)

10:41pm
-> Changed to ***absolute sorting*** to check if that would give better results. Realised this would mix up negative and positively correlated values
-> Looked at correlation values again. They are all weakly correlated.
-> Ideated that it ***must be a non-linear relationship***! Trying to plot scatter plots to assess which features would be relevant and form distinct quadratic or cubic curves.

Error -> features I'm plotting aren't the one-hot encoded versions. Extracting the new columns

10:56pm
-> None of the plots reveal any useful information. Almost always a mismatch between continuous/discrete.

Training Experimentation

Manual implementation of RFE: starting with all features, gradually removing the weakest features and evaluating

1. Top 14 correlated values (categorical and numerical)

```
selected_columns = ['avg_purchase_value', 'customer_city_City C',
'total_returned_items','in_store_purchases','gender_Male','quantity','payment_method_Cash','transaction
_id','total_items_purchased','website_visits','customer_zip_code','days_since_last_purchase','product_revi
ew_count','store_state_State Z','customer_state_State Z']
```

OUTPUT:

urvashibalasubramaniam@Urvashis-MacBook-Air Kabir_Vohra_Urvashi_Balasubraniam_A1 % python3 retail_task/src/train_model.py

Loading data from: /Users/urvashibalasubramaniam/Documents/GitHub/Kabir_Vohra_Urvashi_Balasubraniam_A1/retail_task/data/training_subset.csv
Data shape: (800000, 15)
Features shape: (800000, 14)
Target shape: (800000,)
Target statistics - Min: 0.00, Max: 1.00, Mean: 0.50
Training Linear Regression model...
Training samples: 800000, Features: 14
Learning rate: 0.01, Max iterations: 2000
Regularization (L2): 0.0
Iteration 0: Loss = 1.000025
Converged after 32 iterations
Training completed. Final loss: 0.999966
TRAINING RESULTS
Final MSE: 0.08
Final RMSE: 0.29
Final MAE: 0.25
R-squared: 0.0000
Saving model to: /Users/urvashibalasubramaniam/Documents/GitHub/Kabir_Vohra_Urvashi_Balasubraniam_A1/retail_task/models/linear_regression_baseline.pkl

Model saved successfully!

INSIGHTS:

- Final loss is 0.999966 -> far too high
- R^2 is 0 -> exact same error as simply predicting the mean of the target variable. Model has absolutely no predictive power.
- 32 iterations is also really low for a model with 80k samples

The feature sample is far too small to capture the 149 different relevant columns!

2. All features minus bottom 20 features (by correlation)

- Took features in absolute sorted order (please check the corresponding section for the sorted features)
- Removed bottom 20 (least correlated features) from the dataset

OUTPUT:
Data shape: (800000, 128)
Features shape: (800000, 127)
Target shape: (800000,)
Target statistics - Min: 0.00, Max: 1.00, Mean: 0.50
Training Linear Regression model...
Training samples: 800000, Features: 127
Learning rate: 0.01, Max iterations: 2000
Regularization (L2): 0.0
Iteration 0: Loss = 1.000144
Converged after 56 iterations
Training completed. Final loss: 0.999897
TRAINING RESULTS
Final MSE: 0.09
Final RMSE: 0.30
Final MAE: 0.26
R-squared: 0.0001
Saving model to:
/Users/urvashibalasubramaniam/Documents/GitHub/Kabir_Vohra_Urvashi_Balasubraniam_A1/retail_task/models/linear_regression_baseline.pkl

INSIGHTS:

1. **Incredibly low predictive power** (R squared is almost zero, almost as good as just predicting the average of the target variable for every data point)
2. **High final loss**
3. **56 iterations in gradient descent:** gradients became very small or zero almost immediately, caused the training process to stop prematurely before the model could find a better solution

Thought:

Realised why the model was performing poorly – sorting by absolute values means I'm mixing negative and positive correlation values and eliminating the wrong features in the wrong order!

Now separating the features into positively and negatively correlated and then sorting again and testing separately.

***

Removed absolute sorting and tried again

Thought:

Correlation values are extremely low. Python always tests the linear relationship between two features – perhaps the model isn't linear?

Testing for **quadratic and cubic curves** using scatter plots of every relevant feature with avg_purchase_value

Reducing the dataset: choosing features with educated guesses on their relevance, for the one hot encoded variety:

age
membership_years
number_of_children
transaction_id
product_id
quantity
unit_price
discount_applied
transaction_hour
week_of_year
month_of_year
avg_purchase_value
avg_discount_used
online_purchases
in_store_purchases
avg_items_per_transaction
avg_transaction_value
total_returned_items
total_returned_value

total_sales
total_transactions
total_items_purchased
total_discounts_received
avg_spent_per_category
max_single_purchase_value
min_single_purchase_value
product_rating
product_review_count
product_stock
product_return_rate
product_weight
product_shelf_life
promotion_id
customer_zip_code
store_zip_code
distance_to_store
customer_support_calls
website_visits
days_since_last_purchase
promotion_time_period
gender_Female
gender_Male
gender_Other
income_bracket_High
income_bracket_Low
income_bracket_Medium
loyalty_program_No
loyalty_program_Yes
churned_No
churned_Yes
marital_status_Divorced
marital_status_Married
marital_status_Single
education_level_Bachelor's
education_level_High School
education_level_Master's
education_level_PhD
occupation_Employed
occupation_Retired
occupation_Self-Employed
occupation_Unemployed
product_category_Clothing
product_category_Electronics
product_category_Furniture

product_category_Groceries
product_category_Toys
payment_method_Cash
payment_method_Credit Card
payment_method_Debit Card
payment_method_Mobile Payment
store_location_Location A
store_location_Location B
store_location_Location C
store_location_Location D
day_of_week_Friday
day_of_week_Monday
day_of_week_Saturday
day_of_week_Sunday
day_of_week_Thursday
day_of_week_Tuesday
day_of_week_Wednesday
purchase_frequency_Daily
purchase_frequency_Monthly
purchase_frequency_Weekly
purchase_frequency_Yearly
preferred_store_Location A
preferred_store_Location B
preferred_store_Location C
preferred_store_Location D
product_name_Product A
product_name_Product B
product_name_Product C
product_name_Product D
product_brand_Brand X
product_brand_Brand Y
product_brand_Brand Z
product_size_Large
product_size_Medium
product_size_Small
product_color_Black
product_color_Blue
product_color_Green
product_color_Red
product_color_White
product_material_Glass
product_material_Metal
product_material_Plastic
product_material_Wood
promotion_type_20% Off

promotion_type_Buy One Get One Free
promotion_type_Flash Sale
promotion_effectiveness_High
promotion_effectiveness_Low
promotion_effectiveness_Medium
promotion_channel_In-store
promotion_channel_Online
promotion_channel_Social Media
promotion_target_audience_New Customers
promotion_target_audience_Returning Customers
customer_city_City A
customer_city_City B
customer_city_City C
customer_city_City D
customer_state_State X
customer_state_State Y
customer_state_State Z
store_city_City A
store_city_City B
store_city_City C
store_city_City D
store_state_State X
store_state_State Y
store_state_State Z
holiday_season_No
holiday_season_Yes
season_Fall
season_Spring
season_Summer
season_Winter
weekend_No
weekend_Yes
email_subscriptions_No
email_subscriptions_Yes
app_usage_High
app_usage_Low
app_usage_Medium
social_media_engagement_High
social_media_engagement_Low
social_media_engagement_Medium

3. Attempted with non-absolute sorting

- Results were still poor as features were very weakly correlated with the target

4. Attempted Quadratic Model

- Rewrote train_model.py for a quadratic model
- Did not get robust results

-> Tried modelling just avg_purchase_value to understand its shape. Kernel crashed three times.

Going with quadratic model for now.

Challenges Faced

1. **Kernel crashes during one-hot encoding**
    - Encoding caused the notebook to crash repeatedly (taking >40 seconds and timing out).
    - Root cause: extremely high-cardinality columns (IDs, dates with ~800k unique values) being unnecessarily encoded.
2. **Memory inefficiency with transformations**
    - Converting sparse outputs to dense arrays (`toarray()`) consumed too much memory.
    - Required experimenting with `sparse_output=True` vs. `False` to balance efficiency.
3. **Unique ID columns inflating dimensionality**
    - Columns like transaction IDs and dates had nearly all unique values, making them meaningless for modeling.
    - Had to design rules for feature removal or transformation.
4. **Redundant date features**
    - Multiple date fields provided little or no new information (e.g., `transaction_date`, `last_purchase_date`).

    - Needed careful evaluation and transformation (e.g., deriving `promotion_time_period`).

5. **Difficulty correlating categorical variables with target**
    - `df.corr()` only worked with numerical features.
    - Tried Chi-Square tests but realized they only apply to categorical–categorical comparisons, not with continuous targets.
6. **Testing alternative correlation approaches**
    - Point biserial correlation only worked for binary categories.
    - ANOVA considered for multi-category variables.
    - Ultimately had to fall back on one-hot encoding + correlation.
7. **Redundant testing**
    - Realized after the fact that `corr()` on one-hot encoded data already worked, so some earlier correlation tests had been duplicated unnecessarily.
8. **Weak feature correlations**
    - Both numerical and categorical correlations with the target were extremely weak.
    - Made feature selection and model building more challenging, as linear trends were not apparent.
9. **Discrete vs. continuous mismatch in plots**
    - Scatter plots between continuous target and discrete predictors produced "striped" patterns that were hard to interpret.
10. **Unclear relationships in data**

- Even after visualizations, no meaningful linear or quadratic/cubic patterns emerged.
- Suggested the problem may require non-linear modeling approaches.

Correlated Features (Absolute Value) Sorted Highest to Lowest

['customer_support_calls',
'total_transactions',
'customer_city_City C',
'customer_state_State Z',
'total_returned_items',
'avg_discount_used',
'in_store_purchases',
'product_rating',
'distance_to_store',
'gender_Male',
'store_state_State Z',
'quantity',
'online_purchases',
'payment_method_Debit Card',
'day_of_week_Sunday',
'product_return_rate',
'payment_method_Cash',
'promotion_effectiveness_High',
'membership_years',
'day_of_week_Saturday',
'transaction_id',
'total_items_purchased',
'loyalty_program_No',
'loyalty_program_Yes',
'promotion_effectiveness_Medium',
'store_state_State Y',
'preferred_store_Location D',
'customer_city_City D',
'gender_Other',
'website_visits',
'email_subscriptions_No',
'email_subscriptions_Yes',
'product_material_Wood',
'holiday_season_No',
'holiday_season_Yes',
'min_single_purchase_value',
'number_of_children',
'day_of_week_Monday',
'total_discounts_received',
'education_level_High School',
'discount_applied',
'season_Fall',
'customer_state_State X',
'customer_zip_code',
'purchase_frequency_Monthly',
'customer_city_City B',

'customer_state_State Y',
'month_of_year',
'preferred_store_Location C',
'store_city_City D',
'product_material_Plastic',
'payment_method_Mobile Payment',
'purchase_frequency_Yearly',
'day_of_week_Tuesday',
'payment_method_Credit Card',
'transaction_hour',
'product_material_Metal',
'education_level_PhD',
'promotion_time_period',
'season_Winter',
'store_city_City A',
'days_since_last_purchase',
'promotion_channel_Online',
'product_review_count',
'weekend_Yes',
'weekend_No',
'week_of_year',
'unit_price',
'day_of_week_Wednesday',
'promotion_type_Buy One Get One Free',
'promotion_type_20% Off',
'gender_Female',
'product_weight',
'marital_status_Divorced',
'promotion_id',
'product_material_Glass',
'product_brand_Brand Z',
'income_bracket_Medium',
'occupation_Unemployed',
'store_state_State X',
'promotion_channel_In-store',
'preferred_store_Location B',
'product_category_Clothing',
'store_location_Location C',
'income_bracket_High',
'social_media_engagement_Low',
'product_size_Small',
'product_name_Product D',
'app_usage_Low',
'product_stock',
'total_sales',
'marital_status_Married',
'app_usage_Medium',
'education_level_Master\'s',
'store_city_City B',

'product_brand_Brand Y',
'product_name_Product A',
'day_of_week_Friday',
'occupation_Employed',
'product_color_Black',
'product_category_Furniture',
'social_media_engagement_High',
'store_location_Location D',
'total_returned_value',
'product_size_Large',
'product_color_Blue',
'store_city_City C',
'store_location_Location B',
'preferred_store_Location A',
'promotion_channel_Social Media',
'product_brand_Brand X',
'marital_status_Single',
'avg_transaction_value',
'promotion_effectiveness_Low',
'product_color_White',
'occupation_Self-Employed',
'age',
'education_level_Bachelor's',
'customer_city_City A',
'social_media_engagement_Medium',
'product_size_Medium',
'avg_spent_per_category',
'product_shelf_life',
'season_Summer',
'store_zip_code',
'product_color_Red',
'product_category_Electronics',
'product_name_Product B',
'season_Spring',
'max_single_purchase_value',
'purchase_frequency_Daily',
'day_of_week_Thursday',
'store_location_Location A',
'churned_No',
'churned_Yes',
'app_usage_High',
'income_bracket_Low',
'product_category_Toys',
'purchase_frequency_Weekly',
'product_category_Groceries',
'avg_items_per_transaction',
'product_color_Green',
'promotion_target_audience_New Customers',
'promotion_target_audience_Returning Customers',

'product_name_Product C',
'occupation_Retired',
'product_id',
'promotion_type_Flash Sale']

Numerical Features Sorted Highest to Lowest By Absolute Value of Correlation

Absolute Sorted Numerical Correlations List

avg_purchase_value 1.000000
customer_support_calls -0.002653
total_transactions -0.002650
total_returned_items 0.002510
avg_discount_used -0.002486
in_store_purchases 0.002482
product_rating -0.002480
distance_to_store -0.002428
quantity 0.002272
online_purchases -0.002129
product_return_rate -0.002060
membership_years -0.001789
transaction_id 0.001726
total_items_purchased 0.001661
website_visits 0.001527
min_single_purchase_value -0.001418
number_of_children -0.001362
total_discounts_received -0.001351
discount_applied -0.001325
customer_zip_code 0.001292
month_of_year -0.001189
transaction_hour -0.001081
days_since_last_purchase 0.000979
product_review_count 0.000955
week_of_year 0.000858
unit_price -0.000848
product_weight -0.000790
promotion_id -0.000737
product_stock 0.000556
total_sales -0.000511
total_returned_value -0.000407
avg_transaction_value 0.000262
age 0.000226
avg_spent_per_category -0.000198
product_shelf_life 0.000179
store_zip_code 0.000169
max_single_purchase_value 0.000106

avg_items_per_transaction -0.000041
product_id -0.000019
39 originally numerical features

Explanation

-> The relationship between the predictor variables and the target feature (`avg_purchase_value`) turned out to be really weak overall.

-> Correlation testing revealed that both numerical and categorical predictors exhibited very low correlation values with the target, typically in the range of 0.001–0.002. This indicates that no single variable strongly determines purchase value.

-> Among numerical variables, `total_returned_items`, `in_store_purchases`, and `quantity` showed the highest correlations, though still very weak.

-> Among categorical variables, `gender_Male`, `customer_city_City C`, `payment_method_Cash`, and `loyalty_program_No` were the most relevant.

These findings suggest that while these factors may contribute slightly, they cannot independently explain variations in purchase value.

-> Visual inspections reinforced this insight: scatter plots showed striped or overlapping distributions due to the discrete nature of predictors against a continuous target, and box plots of categorical features displayed overlapping ranges.

-> Together, these results imply that the dataset likely exhibits non-linear interactions, where the combined effect of multiple predictors may be more meaningful than any single variable in isolation.

Therefore, while the linear correlation approach provided some useful signals, it is limited in explaining the target.

The insights point toward the need for models capable of capturing non-linear patterns and interactions—such as decision trees or ensemble methods—to better reflect the relationships in the data. Even the quadratic model had non-robust results, further emphasising the need for a more advanced modelling technique.

Conclusion

Analysis of the retail dataset demonstrated that the relationship between predictor variables and the target feature (`avg_purchase_value`) is generally weak.

Correlation testing, both for numerical and categorical variables (using Pearson and Point-Biserial respectively), consistently produced very low values, suggesting that no single feature exerts a dominant influence on purchase value.

Visualizations further confirmed this, as scatter plots and box plots revealed little to no clear linear trends.

The findings highlight that while the dataset is rich in categorical and transactional detail, ***most variables are only marginally informative in isolation***. This suggests to me that `avg_purchase_value` is influenced by more complex, non-linear interactions among features rather than simple direct relationships, and thus requires more complex modelling techniques.

Possible improvements:

1. **Feature Engineering** – Create new interaction features (e.g., `spend per visit`, `average discount per item`) or higher-level aggregates (e.g., customer lifetime value).

2. **Non-Linear Models** – Move beyond simple correlation and linear regression, testing tree-based models (Random Forest, XGBoost) or neural networks that can capture interaction effects.

3. **Temporal Features** – Incorporate seasonality or lagged purchase behavior more explicitly rather than discarding date columns entirely.

4. **Clustering / Segmentation** – Explore unsupervised learning to segment customers by purchase patterns, which may reveal hidden group-level trends not visible in global correlations.

Results

Please refer to jupyter notebook for retail task for the associated graphs, as copy-pasting them here would make it difficult to read and understand. There are three sections on visualisations in the notebook, including heatmaps, scatter plots and graphs. Thank you!

Resources

1. Implementing linear regression model from scratch – https://www.youtube.com/watch?v=VmbA0pi2cRQ&ab_channel=NeuralNine
2. **Explored point biserial correlation method –** but this only works for binary categorical variables.
3. https://datascientest.com/en/calculate-correlation-between-two-variables-how-do-you-measure-dependence. Instead using ANOVA, as there are several columns with n>2 unique values.
4. Extract date info using this method, so we don't have to delete the features– https://medium.com/@paghadalsneh/handling-date-and-time-data-in-machine-learning-a-comprehensive-guide-5d30141cbfec
5. Insights from https://koshurai.medium.com/point-biserial-correlation-32440e373718 on point biserial
6. Krish Naik Machine Learning https://www.youtube.com/watch?v=7uwa9aPbBRU&list=PLTDARY42LDV7WGmlzZtY-w9pemyPrKNUZ&ab_channel=KrishNaikHindi
7. RowZero
8. Pandas Documentation https://pandas.pydata.org/docs/
9. Heatmaps https://www.geeksforgeeks.org/python/how-to-draw-2d-heatmap-using-matplotlib-in-python/