The key steps of Machine Learning model development: Example of an end-to-end Machine Learning pipeline

Business goal and machine learning solution

Black Friday is one of the days with the largest volume of commercial activity. Customers are flocking to take advantage of good deals, and businesses are trying to sell as many products as possible to increase their profit. Therefore, it is particularly important that any business needs to get the pricing right for this particular day - if the price is set too low - the business will have suboptimal profits, and if too high, customers will just simply buy from other competitors which are offering better deals.

What is even more challenging for setting the correct price, is that not only do the discounts take a wide range of values, they are also seemingly arbitrary. Some goods can come off to sell at a massive discount, and some goods sell at practically the same price as before. Customer purchasing behaviour cannot be easily predicted as well, as each customer is different, with different price sensitivity.

Therefore, a machine learning approach is needed to determine an appropriate price for a given product, taking advantage of user information among other factors. Here, combining with product data, we use sales data which tracks customers' purchases and their amount, alongside their demographics such as gender, age, marital status etc, to train a machine learning model to suggest a price for a given product and customer. Personalised pricing strategy and offers could then be used as a result, which will drive higher profits.

It's worth pointing out that customer demographics data used in this demo is for the benefit of understanding model performance and potential bias. Bias is addressed in the Fairness analysis section (3.2.3.8) and potential mitigation methods are discussed accordingly. While personalised pricing is a complex and debated topic in terms of pricing fairness, we aim to propose alternative angles to improve AI fairness and achieve profit enhancement at the same time.

Data Exploration

Decision summary:

- **Model type: ml problem framing: regression**
- **Feature engineering to be focused on key "discount"-related feature reconstruction.**
- **Under/over-represented groups were observed across different demographic features. Thus fairness evaluation will be focused on related groups.**
- **To avoid large errors, we select the model architecture, based on the model's RMSE performance.**

The first step to any machine learning problem is the data, which in this case, is the Black Friday dataset which is available on Kaggle. This dataset contains the purchase history of different customers, which contains information on both the products (e.g. product category and ID), and the customers (gender, age etc) , alongside with the purchase amount. Here, we describe the steps taken to explore the data to understand it, along with any insights we have gained through this process.

Tools used and methodology

The data exploration for this method was done in Google Cloud, on their platform built for Machine Learning, Vertex AI. Specifically, the data exploration was done in Vertex AI Workbench, where notebooks could be created to run arbitrary python code snippets. The graphs generated in this section were all generated with custom code, by using visualisation libraries installed in python (such as seaborn and matplotlib).

Data overview

The Black Friday dataset on Kaggle contains a list of purchases from customers from a company. It is provided in a csv format, with a train set consisting of 25MB and around 550,000 records. Because the test set provided omitted the target variable, we excluded the test set for we can't run evaluation on it. Therefore the total data available is 550,000 records.

Demographic analysis

An initial look at the dataset suggests that a significant proportion of the data contains demographic information, such as gender, age, marital status etc. We have to be extra careful when developing machine learning models with demographic data, as the model will learn any bias in the data (if present), thus enforcing and propagating harmful systemic biases which are present in the real world

- **Gender**

There is clear bias in the dataset. Not only do women purchase significantly lower amounts, they also appear in fewer numbers in the dataset (fig.1) - there are over 3 times as many men as there are women.

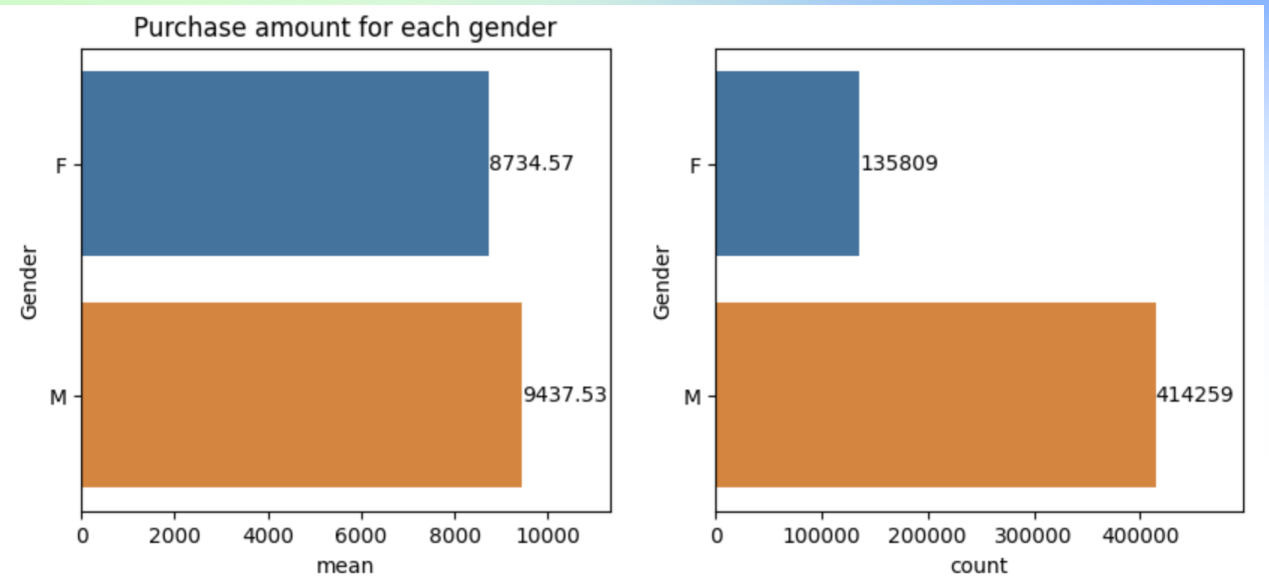


Fig.1 Data distribution: Gender

- **Occupation**

Although there is comparatively less variation compared to Gender, some occupations, such as 9 and 17, produce abnormally high or low values for purchases. However, the problem lies with the distribution of the occupations - some are significantly underrepresented - Occupation 8 has 46 times less records compared to the maximum (fig.2).

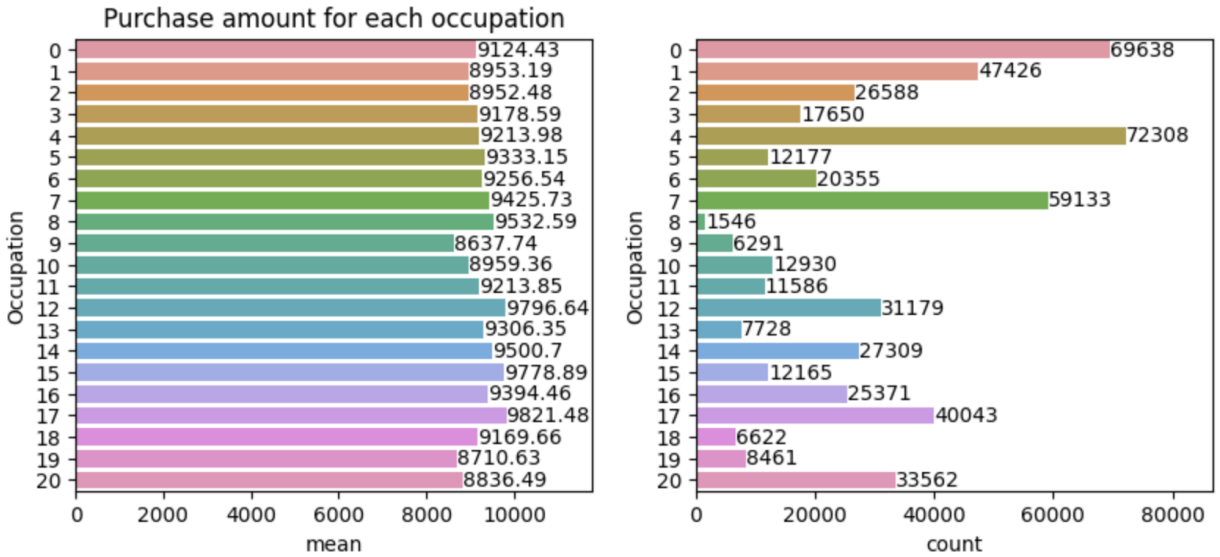


Fig.2 Data distribution: Occupation

- **Age**

There is a minor trend present that older customers tend to spend more. This tends to

corroborate with the intuition that older customers have more available disposable income, thus willing to spend more. However, the effect this has is small - the difference of the purchasing amount between the highest and lowest age groups is less than the difference of the genders. The distribution of the ages is also not uniform, with a notable peak at the 26-35 age group (fig.3).

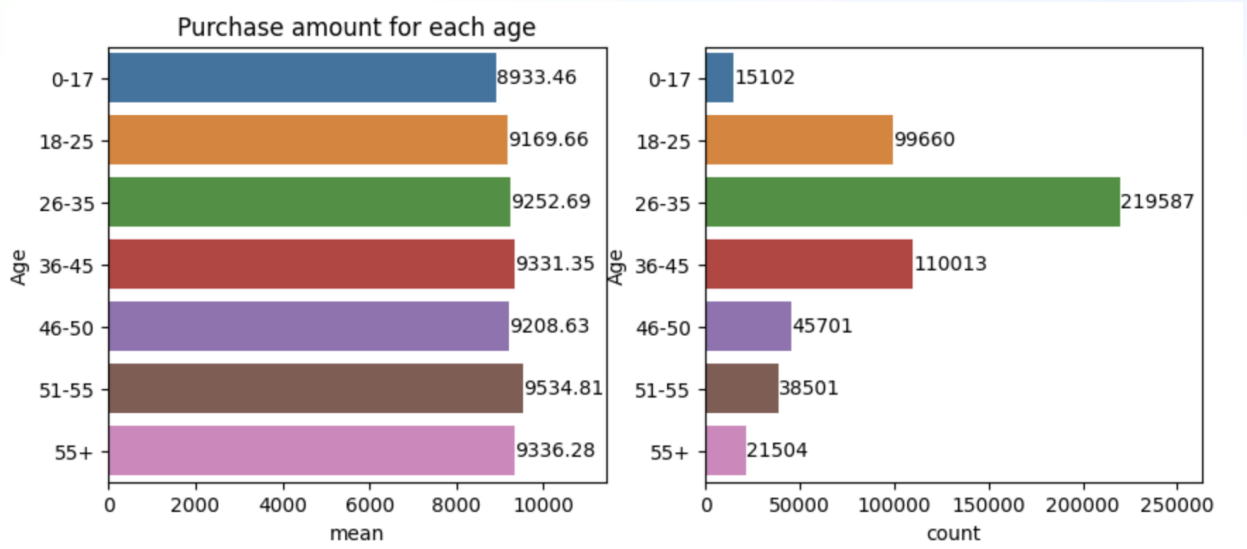


Fig.3 Data distribution: Age

- **Marital status**

Unlike the other demographic groups, marital status has negligible effect on the purchasing value for the customers. Although there are more single people than married people in the dataset, the skew is not large enough to cause problems as both groups are large enough to be representative (fig.4).

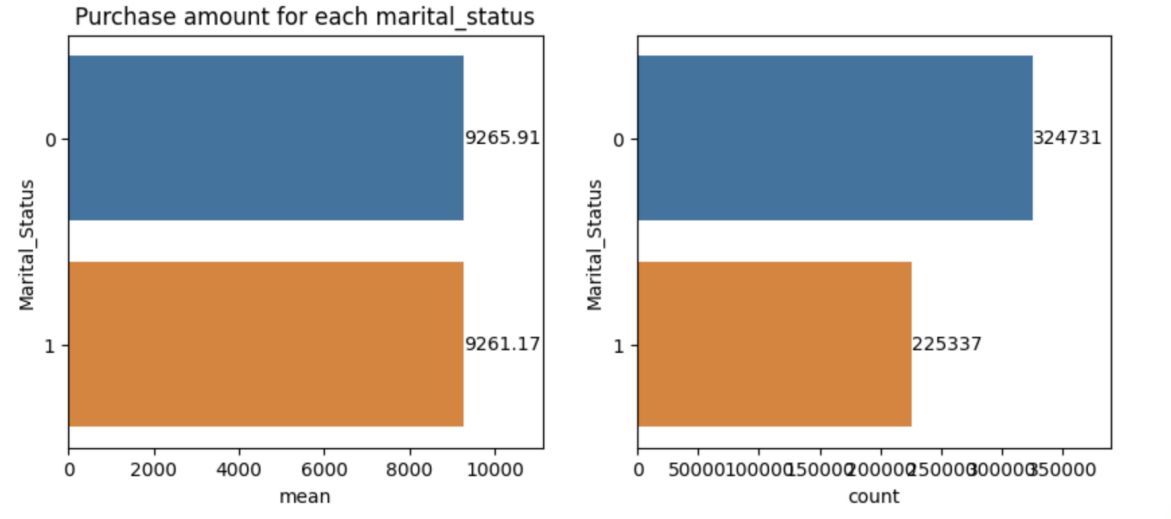


Fig.4 Data distribution: Marital status

Target variable distribution analysis

As we are predicting the purchase amount, it is important to analyse it to gain any potential insights. Unexpectedly, the distribution of the purchase amount is not smooth (fig.5). With thousands of products available with many purchases for each, any noise and peaks would theoretically be smoothed out.

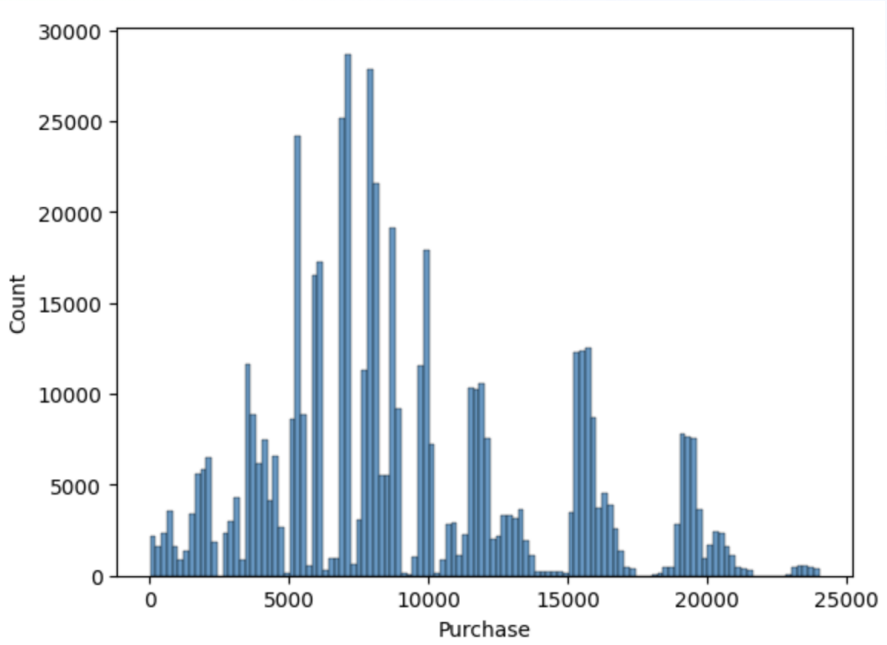


Fig.5 Data distribution: Purchase amount

We hypothesise that the distribution for a given product is very peaky, and a few large actors are contributing to these larger peaks.

Taking a look into distribution of purchase for a given product, instead of seeing one peak as expected, we saw multiple peaks - 5 of them in fact (fig.6). From this, it is abundantly clear that the prices follow 5 distinct discount offers, with some customers paying up to 5 times the price as other customers. The distribution of the proportions of the discounts vary from product to product, with some products having significantly more people buying at full price, while others being extremely generous in offering discounts.

This discount factor is a key predictor for the purchase price, as correctly predicting it would narrow the range of purchase prices significantly. However, this feature is not available during use of the machine learning model - if it did, then it would defeat the entire purpose of this machine learning model. The logical next step is, that during feature engineering, we would want to reconstruct the discount factor.

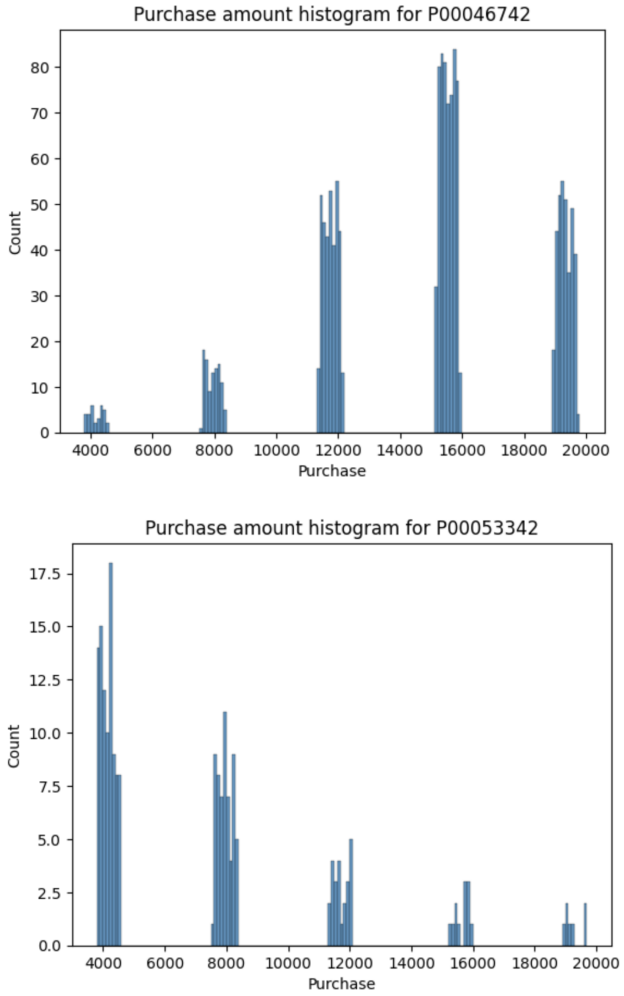


Fig.6 Purchase amount for individual product

Customer behaviour analysis

Next, we look at the purchase value, from the perspective of a customer. We first look at the histogram for the average purchase value for a given customer. Unlike the distribution for the price, it is extremely smooth, resembling the shape of a bell curve (fig.7).

We then analyse the number of items purchased per customer. All the customers in the dataset have made multiple purchases, with the minimum of 6 purchases per customer. The distribution is extremely long tailed, with some customers buying over 1000 purchases (out of view of the graph).

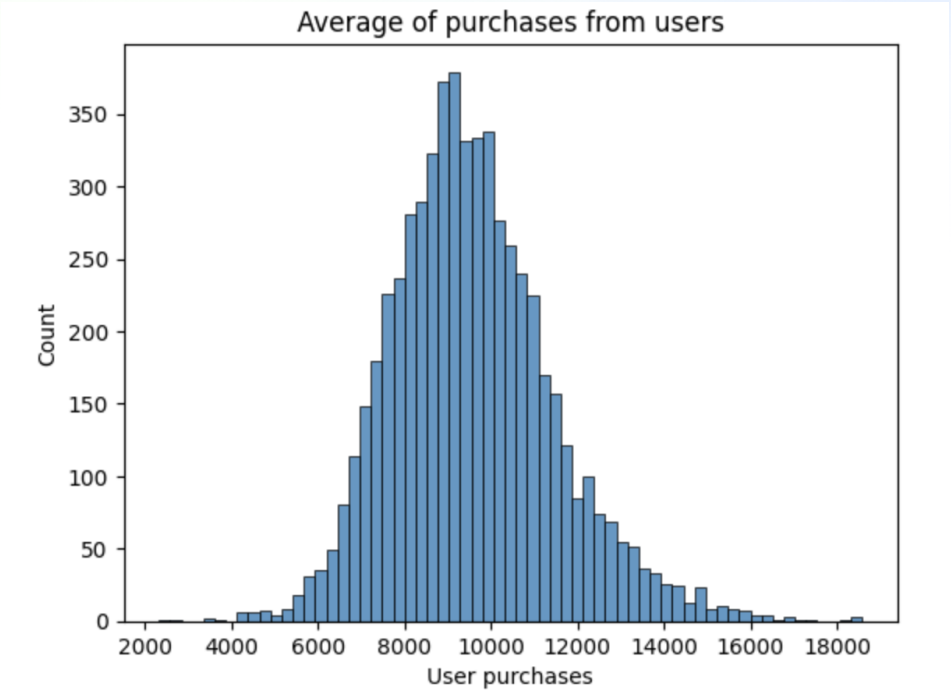


Fig.7 Purchase amount distribution aggregated on users

Feature Engineering

From our data analysis, we found out that the most pertinent feature is the rate at which the product is discounted, where some customers could be charged 5 times higher prices than others. This corroborates with our intuition that the deals offered in Black Friday are seemingly arbitrary. Nonetheless, we will attempt to reconstruct this factor during feature engineering.

A deal is made when the customer believes that they have secured a low enough price, and the business believes that the price charged is high enough to make significant profits. This obviously will vary from customer to customer, as some are more price sensitive than others. The same could be said for businesses, as for the products with more flexible manufacturing processes or cheaper storage, businesses are more willing to shift larger volumes at lower prices as they could more easily adjust production to meet demand.

Therefore, we would need to make use of historical data to estimate a customer's tendency to receive discounts, and the product's tendency to produce discounts. As most customers are recurring customers with a mean of 90 purchases, there is sufficient

data to provide a reasonable estimate for their tendency to receive discounts. The same could be said for the products, as each product has a mean of an even higher 150 records.

For this, tables which aggregate historical data for both customers and products were created. The table contains the mean and the standard deviation of the discount rate, for each product and customer. These tables were then joined to append this data to the records in the training.

With this, the features to be used in the machine learning model are as such:

- Age
- City_Category
- Gender
- Marital_Status
- Occupation
- Stay_In_Current_City_Years
- Maximum product price
- Average discount offered for product
- Number of products sold (for this product)
- Std.dev of product discount
- Average discount bought by customer
- Number of products bought (for this customer)
- Std.dev of customer discounts

Preprocessing and the data pipeline

As the entire dataset is imported in BigQuery, it is the most logical to do all the data preprocessing in BigQuery as well. We would need separate queries to first produce the aggregates, and then join them with the main dataset. The whole data would be processed as follows

1. Split the dataset into test and train datasets
2. Produce aggregates over the train dataset (over products and over customers)
3. Join aggregates with train set to prepare data used for training. Select features
4. Join aggregates with test set and impute missing aggregates by obtaining the mean

During training, this data preprocessing procedure is implemented in the machine learning pipeline via Vertex AI pipelines. This means that the data processing will automatically run every time before the machine learning procedure starts without any manual intervention.

Machine learning model design(s) and selection

Summary: Boosted Tree Regressor was chosen as model architecture because it produced the least Root Mean Squared Error (RMSE). Selection criteria were generated directly from BigQuery ML (BQML) as it's used for rapid prototyping.

Framing the problem as a regression task, we narrowed down possible designs of the model to three options - a simple linear regression model, a deep neural network (DNN) , and a boosted tree regression model.

As BigQuery ML (BQML) could train and evaluate machine learning models with minimal effort, it was used to rapidly prototype and test out different model architectures. A machine learning model was trained for each model architecture using BQML, and returned performance metrics are used to compare with one another.

Though it took the longest to train, the boosted tree has the best performance out of all the models. Therefore, it was selected to be the chosen model architecture.

Model Architecture	Root Mean Squared Error (RMSE)	R^2
Linear regression	2554	0.73
Deep neural network (DNN)	2545	0.73
Boosted tree regressor	2376	0.78

Machine learning model training and development

Once we have decided on the model architecture used, we train the model on **Vertex AI** with optimization. We leverage the **Custom Training** and **Hyperparameter Tuning** components in **Vertex AI Pipeline** to optimise our model. Dataset sampling is done via **BigQuery** and is wrapped in the single **end-to-end Vertex AI Pipeline** as well (fig.8).

Data sampling

As there is no indication of which records are more relevant than other records, i.e. no variable suggesting the recency of data, we sampled **85%** of the data **randomly** as the **Training set** (467,500 records), and the rest **15%** as the independent **Testset** (82,500 records). **Validation set** is subsampled from the total **Training set** during each training run with a default ratio **0.1** (tfdf.keras.GradientBoostedTreesModel, 'validation_ratio').

Model training implementation

To ensure we follow the **Google Cloud best practices** and maximise the reusability and flexibility of our implementation, **we used Vertex AI Pipelines to orchestrate the ML workflow**.

Distribution: We trained a model within a **notebook instance for experimentation** first. Then we **containerized** our custom training code and its dependencies through a python distribution and submitting a **training job to Vertex AI**. In this way our training implementation is also ready for high-performance or distributed training if needed.

Device usage for storage and training: We stored our data and model artefacts in **Bigquery** and **Cloud Storage** for best integration with Vertex AI. As the model size and dataset is not prohibitively large, we selected a simple **n1-standard-4** device for training.

Monitoring: Model monitoring is enabled when deploying the custom model to an endpoint via

console.

Specifically, the pipeline first runs preprocessing on Bigquery, and then uses Bigquery to split into train and test sets. Afterwards, the training job is run, with built in hyperparameter tuning. Once the best model is selected (after hyperparameter tuning), it is uploaded to the model registry, and finally evaluation is done on the test set. This entire process is simply by triggering the pipeline through a single API call.

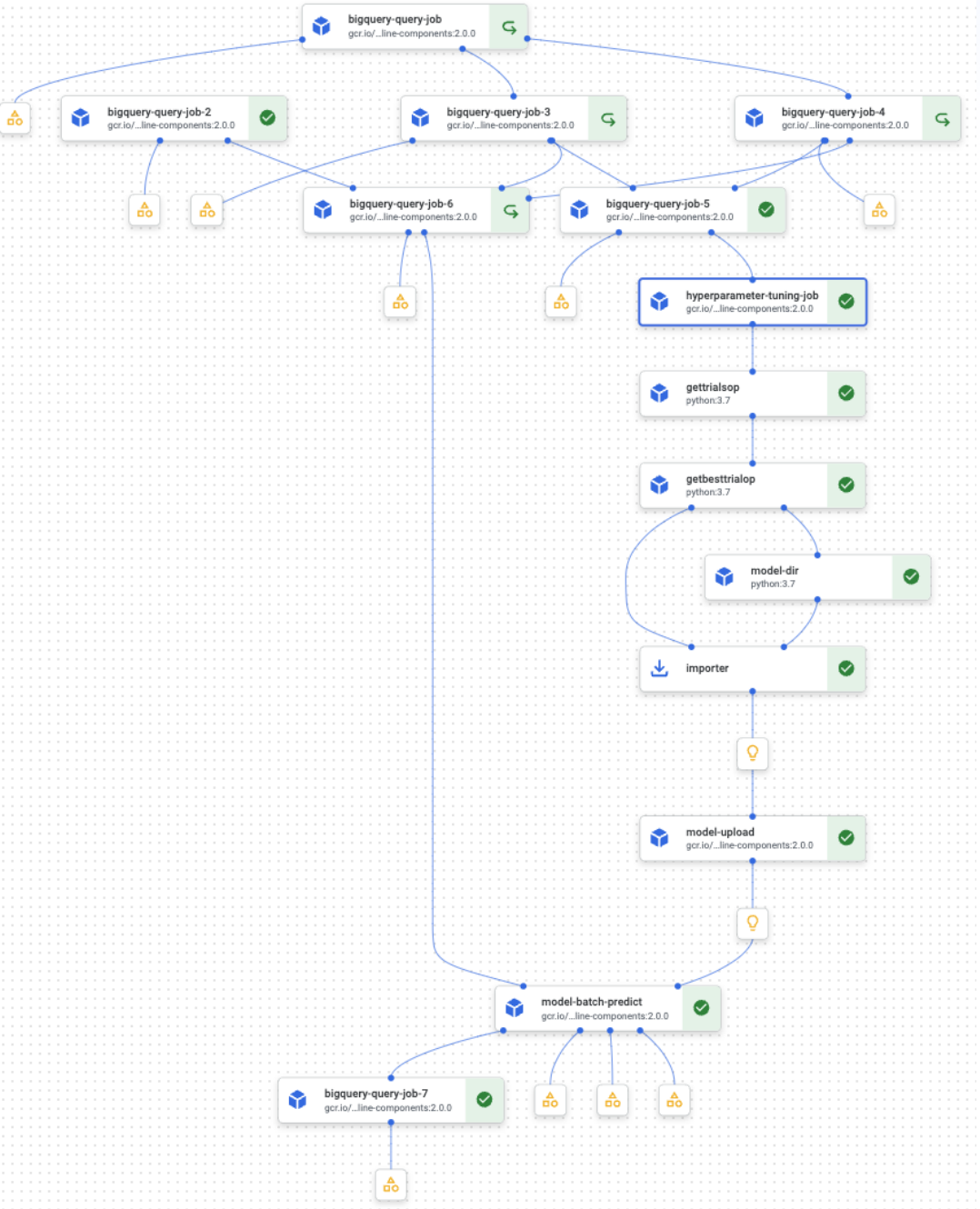


Fig.8 End-to-end Vertex Pipeline Orchestration

Evaluation metric

Squared Error is selected as the loss function as we want to disproportionately penalise large

errors in Purchase amount. This leads to **RMSE** as an effective evaluation metric, and we also compute other metrics such as **MAE** and **MAPE** to give us a holistic view of model performance.

```python
#Train a boosted trees model
model = tfdf.keras.GradientBoostedTreesModel(
   task = tfdf.keras.Task.REGRESSION,
   loss = 'SQUARED_ERROR',
   max_depth = args.depth,
)
hist = model.fit(train_ds)
```

Model performance optimization

Hyperparameter tuning is done using the **Hyperparameter Tuning component** in Vertex AI pipeline (function *HyperparameterTuningJobRunOp*), and is automatically triggered when the data processing component is finished.

Specifically, the optimization target is set to **minimise validation loss**, and the hyperparameter being tuned is the **'depth' of the decision tree**.

```python
#Train model and tune hyperparameters
hptune_job = HyperparameterTuningJobRunOp(
   display_name = 'HPT Test',
   project = PROJECT_ID,
   location = REGION,
   base_output_directory = PIPELINE_ROOT,
   worker_pool_specs = [{
      'machine_spec':{"machine_type": 'n1-standard-4'},
      'replica_count':1,
      'python_package_spec':{
        "executor_image_uri": 'europe-docker.pkg.dev/vertex-ai/training/tf-cpu.2-11.py310:latest',
        "package_uris": ['gs://ml-spec-demo2/dist/trainer-0.1.tar.gz'],
        "python_module": 'trainer.task',
      }
   }],
   study_spec_metrics =  serialize_metrics({'val_loss': 'minimise'}),
   study_spec_parameters = serialize_parameters({
      'depth': hyperparameter_tuning.IntegerParameterSpec(min=4, max=8, scale='linear'),
   }),
   max_trial_count = 2,
   parallel_trial_count = 2,
   study_spec_algorithm = 'ALGORITHM_UNSPECIFIED'
).after(bq_training_data)
```

Optimised model artefact is exported in gcs bucket as automated by our pipeline in the following location:

https://console.cloud.google.com/storage/browser/ml-spec-demo2/custom_pipeline_20230823144928/1/model

Bias and Variance

Bias-variance curve (fig.9) tracks RMSE on the training set (blue) versus the validation set (orange) during training iterations. RMSE dropped effectively during the first stage of training (iteration 0-50), meaning that **Bias** was significantly reduced when the model learned effective weights to fit the data. However, towards the end of training (iteration 150-300) validation loss stayed almost still when the training loss continued to decrease, suggesting the model is overfitting thus **Variance** started to increase. Hence we applied an **early stopping** strategy and selected the best model when the validation loss was not decreased a comparable amount as its divergence from the training loss (iteration 46).

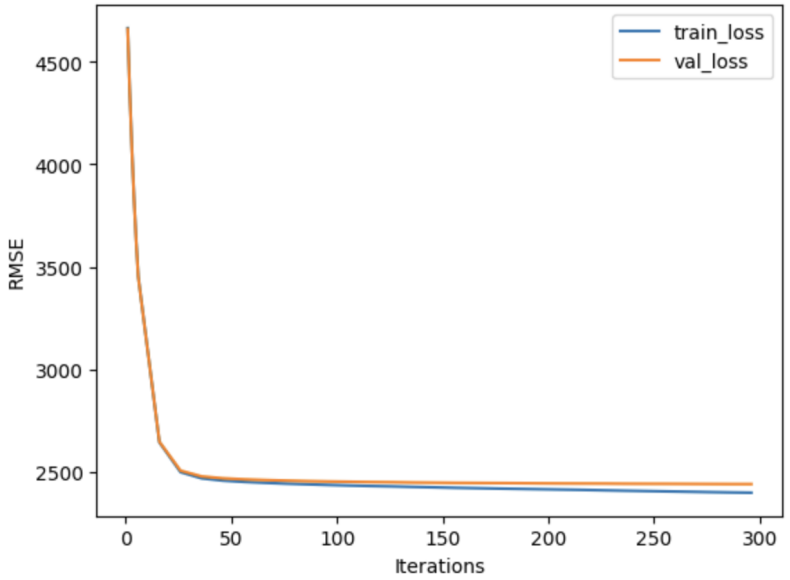


Fig.9 Bia-variance curve for custom training

Machine learning model evaluation

Partner must describe how the machine learning model, post-training, and architectural/hyperparameter optimization performs on an independent test dataset.

Evidence must include records/data (in the whitepaper) of how the machine learning model developed and selected to address the business question performed on an independent test dataset (that reflects the distribution of data that the machine learning model is expected to encounter in a production environment). In addition, code snippets on model testing need to be enumerated.

Independent testset reflecting production environment

As defined in the Data Sampling section, the testset we used here is a random subset (15%) of the total data, thus should represent the overall distribution. Distribution of the testset variables are shown side to side with the total data (i.e. “production data”)

distribution we analysed in the EDA section as evidence (see below figures).

- **Gender**

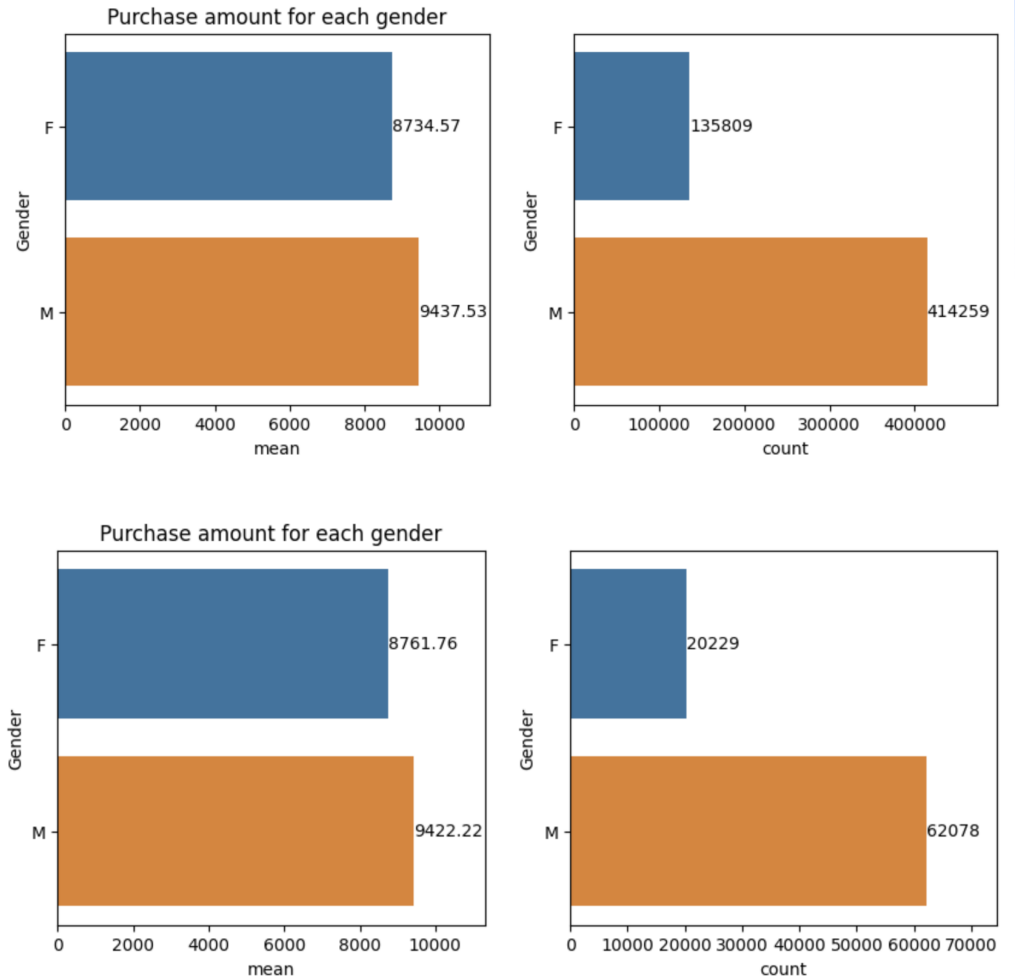
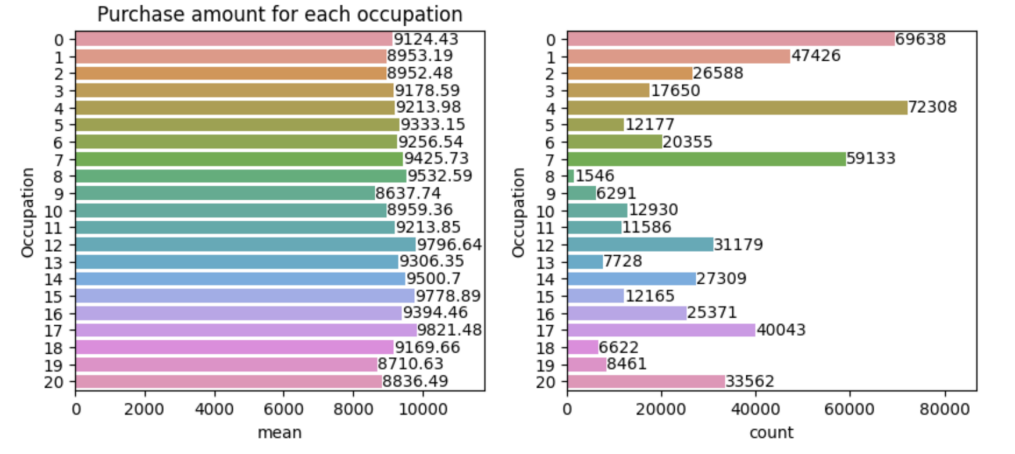


Fig.10 Gender distribution Production (top row) vs. independent Testset (bottom row)

- **Occupation**



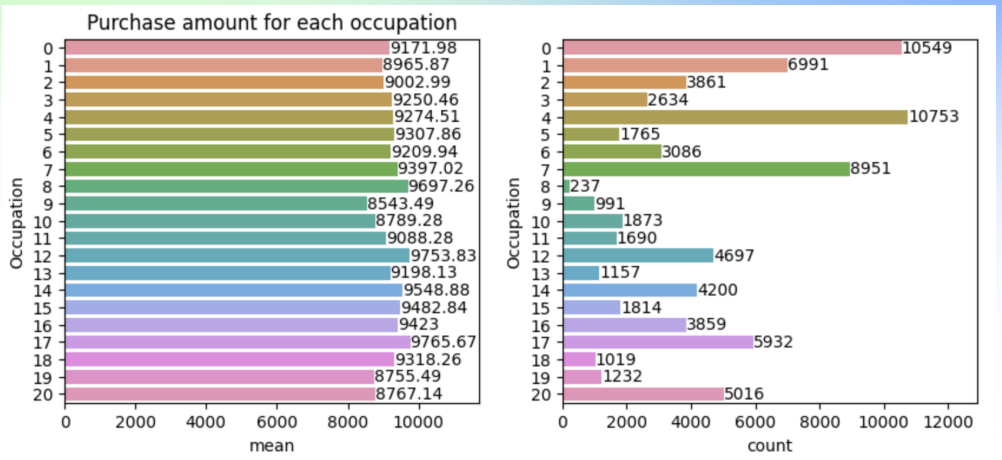


Fig.11 Occupation distribution Production (top row) vs. independent Testset (bottom row)

- **Age**

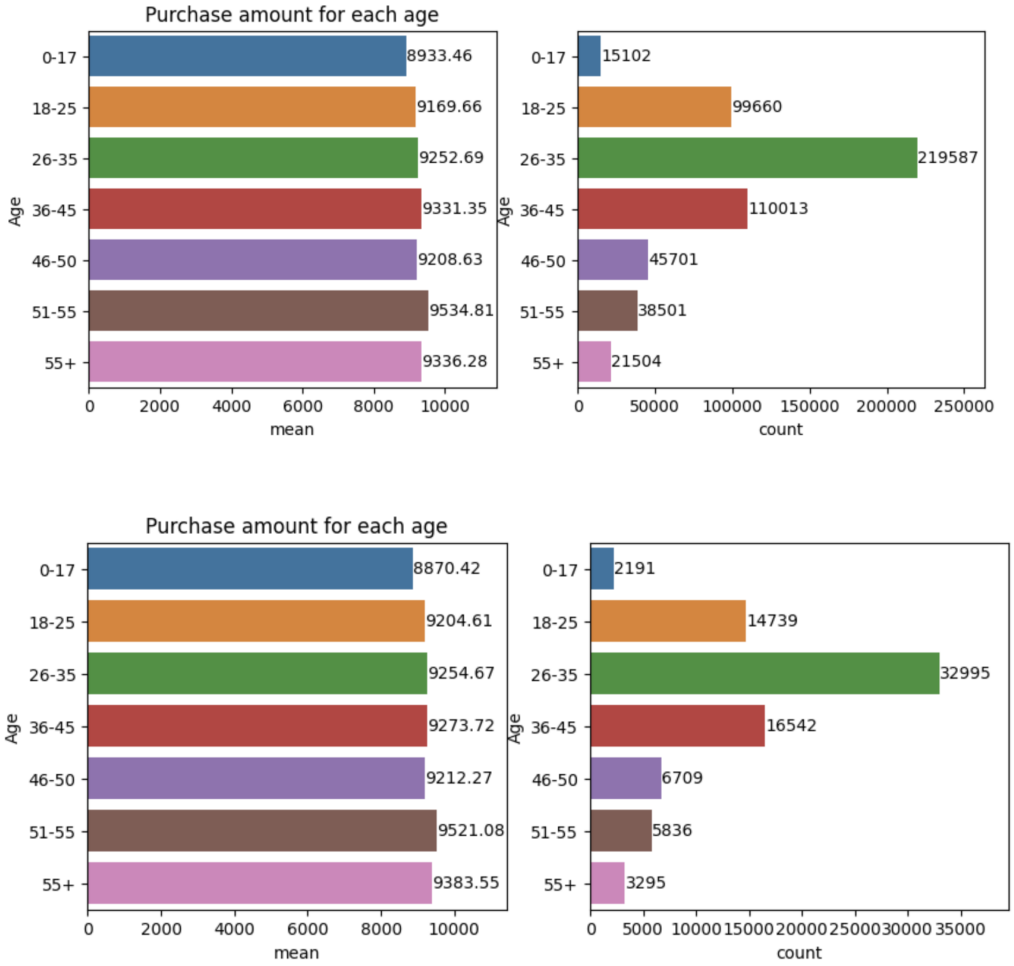


Fig.12 Age distribution Production (top row) vs. independent Testset (bottom row)

- **Marital status**

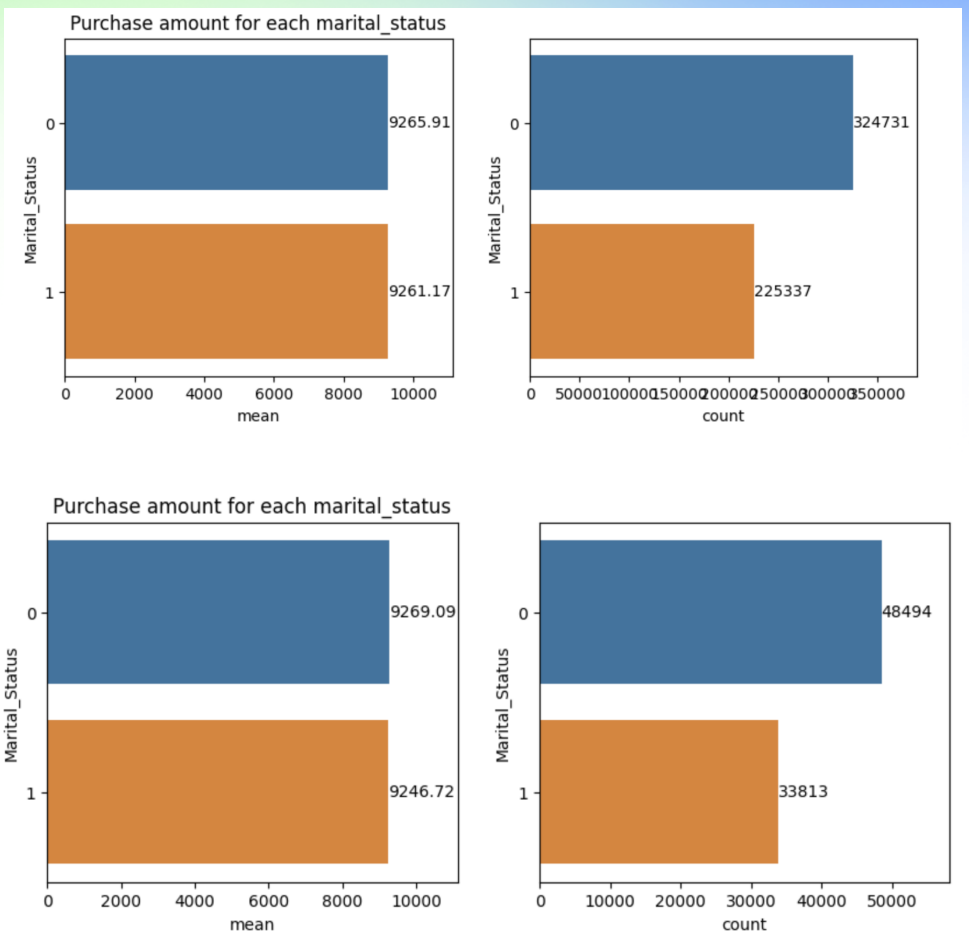
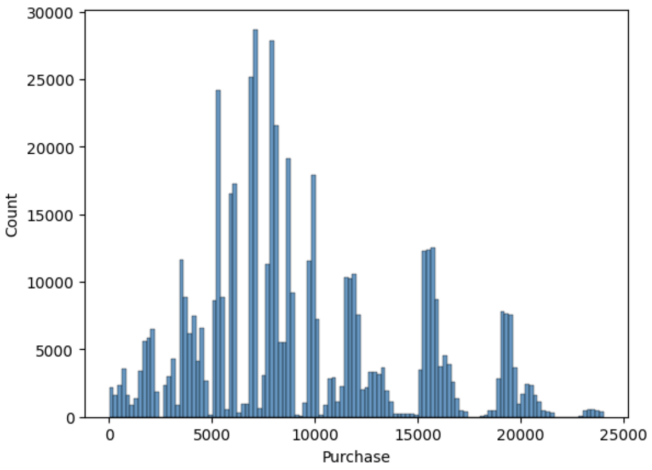


Fig.13 Marital status distribution Production (top row) vs. independent Testset (bottom row)

- **Target variable distribution analysis**



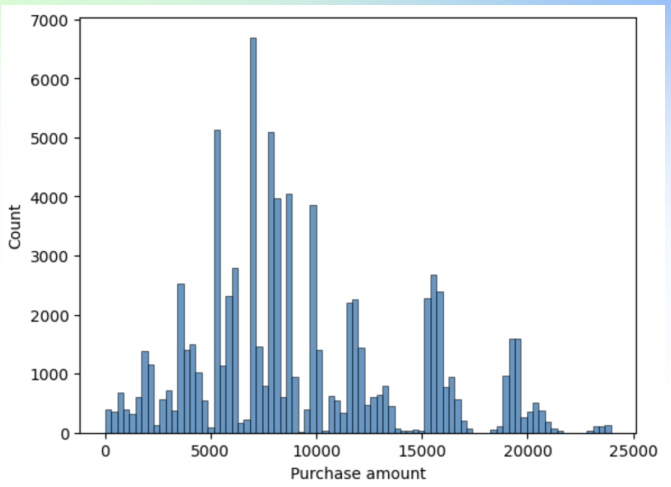


Fig.14 Purchase amount distribution Production (left) vs. independent Testset (right)

- **Customer behaviour analysis**

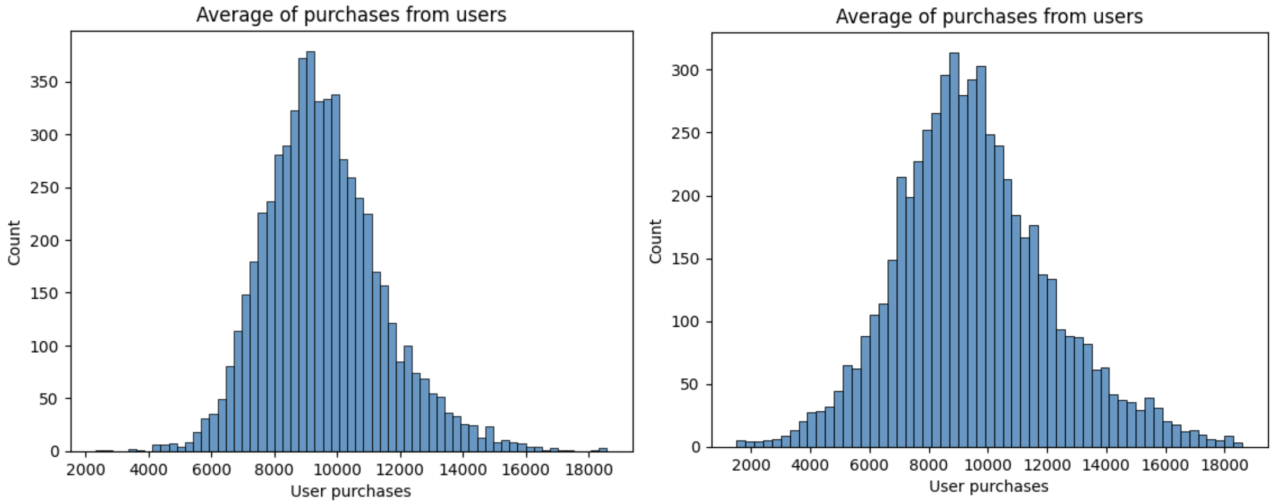


Fig.15 Customer Purchase amount distribution Production (left) vs. independent Testset (right)

Model performance on independent test set

Evaluations were made on the independent test set by first running predictions through it. This is done via the **Batch Prediction** functionality of Vertex AI, which writes the predictions into a **BigQuery table**.

Afterwards, through the use of Bigquery, the predictions and the ground truth are collated, compared, and then **aggregated to produce metrics** as shown in the table below.

Given the extremely arbitrary nature of the discounts, the model was able to predict the purchases fairly accurately with an RMSE of around 2360. Upon simple inspection, most of the predictions were in line with what is expected.

Model Architecture	Root Mean Squared Error (RMSE)	Mean Absolute Error (MAE)	Mean Absolute Percentage Error (MAPE)
Boosted tree regressor	2359	1730	0.2752

Above evaluation steps were wrapped in our **Vertex AI pipeline** as well, code snippet see below:

```python
#Batch prediction
batch_predict = ModelBatchPredictOp(
    project = PROJECT_ID,
    location = REGION,
    job_display_name = 'Predictions_testset',
    model = model_upload.outputs["model"],
    instances_format = 'bigquery',
    predictions_format = 'bigquery',
    bigquery_source_input_uri = f'bq://{PROJECT_ID}.{dataset}.bf_testdata',
    instance_type = 'object',
    excluded_fields = ['Purchase'],
    bigquery_destination_output_uri = f'bq://{PROJECT_ID}.{dataset}.bf_outpred',
    machine_type = 'n1-standard-4',
    max_replica_count = 1
).after(bq_testing_data).after(model_upload)


model_eval = BigqueryQueryJobOp(
    project=PROJECT_ID,
    location='EU',
    query=f"""
    create or replace table {dataset}.bf_evalmetrics as(
    with preds_v_gt as (
      select
        Purchase as gt,
        round(cast(trim(prediction, '[]') as numeric),3) as pred
      from `{dataset}.bf_outpred`
    )
    select 'RMSE' as metric, sqrt(avg(pow(gt-pred,2))) as value
    from preds_v_gt union all
    select 'MAE',avg(abs(gt-pred))
    from preds_v_gt union all
    select 'MAPE',avg(abs(1-pred/gt))
    from preds_v_gt
    )
    """
).after(batch_predict)
```

Fairness analysis

Assessing the fairness of AI systems is an important part to hold AI responsible in real-world production. Specifically for this use case, it's unethical to apply any dynamic

pricing discriminated against demographic characteristics of individual customers. However, on the analytical side, we wanted to include as many features as possible in order to first understand what factors contribute to the purchase amount at each sale. Therefore, we'll first discuss the role of demographic features in theory, then move to address model bias, and finally to the application side where we propose mitigation solutions and alternative business strategies to ensure optimal fairness.

Is demographic feature helpful for predicting purchase amount?

We explored the Explainable AI functionality for our best model so far and the importance of each feature is listed below (fig.16, importance high to low). The top importance goes to *mxp* (maximum product price) and is significantly outweighing other features, meaning the model prediction is largely driven by the max price a customer has paid before, i.e. personal purchase history. Features also come up in the top five importance are *p_avg* (product average discount), *u_avg* (customer average discount), *u_cnt* (number of products bought), and *u_std* (customer discount std), all relating to personal/product history information that were constructed during feature engineer. In contrast, demographic features are of the lowest importance: *Gender, Occupation, City_Category, Stay_In_Current_City_Years, and Marital_Status* are listed in the bottom five, except for *Age* which is of the sixth in importance ranking. Therefore, our conclusion is demographic features in this dataset are of very little importance to the model prediction. Instead, the most important indicator of future purchase amount is a customer's purchase history. Other helpful features include information in product sales history.

In this sense, the model is not likely to heavily bias to certain demographic groups because it's simply not much dependent on demographic features. Still, it's worth examining the model performance directly amongst each demographic subgroup to directly inspect any bias existing.

Feature name	Attribution
mxp	3,516.438
p_avg	865.014
u_avg	675.785
u_cnt	74.96
u_std	59.069
Age	44.167
p_std	33.12
p_cnt	24.864
Gender	22.157
Occupation	20.221
City_Category	19.613
Stay_In_Current_City_Years	7.327
Marital_Status	2.807

Fig.16 Feature importance boosted tree model

Is the model biased to/ discriminate against particular demographic groups?

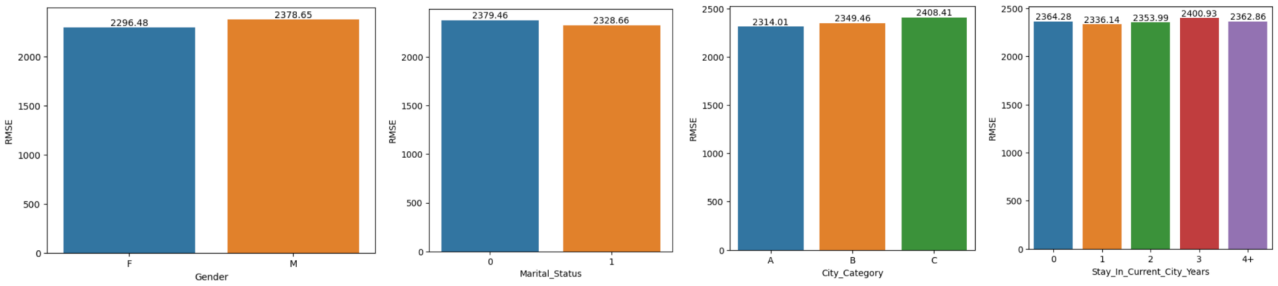
To reveal any existing bias the model could hold against each group, RMSE is calculated for each demographic group as a numeric indicator of model performance. Note that the average RMSE across the total testset is **2359**.

RMSE difference between groups divided by *Gender, Marital_Status, City_Category, and Stay_In_Current_City_Years* is relatively small, with a std of **20-60**, while *Occupation* and *Age* has larger RMSE difference between groups (std **> 100**). **This means the model has a slight tendency to perform better/worse towards certain *Occupation* or *Age* groups than others.**

	Gender	Marital_Status	City_Category	Stay_In_Current_City_Years	Occupation	Age
RMSE std	58.10	35.92	47.68	23.67	128.05	108.78

Model bias in *Occupation* is consistent with our EDA findings that certain occupation groups are extremely underrepresented (e.g. Occupation 8). Therefore high RMSE in Occupation 8 is expected.

Model bias in *Age* groups is more complex and interesting. While high RMSE in the 0-17 age group can be explained by underrepresentation , the model is not performing better in the 18-25 group although more data is available. This could indicate more heterogeneity in purchase behaviour in the 18-25 group. In addition, the model performs better in older age groups although less data is available, suggesting purchase behaviour is more predictable in older age groups.



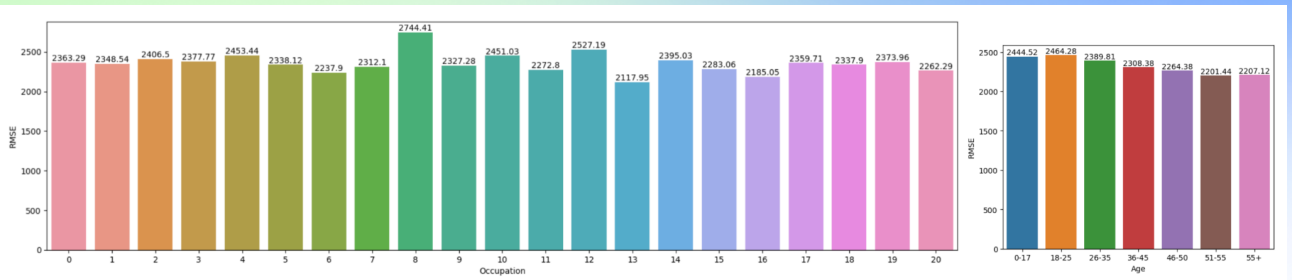


Fig.17 RMSE comparison between demographic subgroups
(Top row: **Gender, Marital_Status, City_Category, Stay_In_Current_City_Years**;
Bottom row: **Occupation, Age**)

Is personalised pricing fair?

Now that we've explored the actual bias in the current model, we did some background research to understand fairness concerns in personalised pricing in real practice.

Personalised pricing can enhance economic efficiency by aligning prices with consumers' willingness to pay. It may benefit consumers by offering discounts based on income or sensitivity to prices, though one could argue it's benefiting certain groups at the sacrifice of the rest. Transparency is another consideration, as hidden or discriminatory practices can be unfair, but informed personalised pricing could decrease customer's willingness to buy. Concerns about data privacy and abuse of personal information also arise. Therefore, we think our fairness solution should aim at a balance between market efficiency and consumer protection.

Fairness solution

Following the above analysis we propose several fairness solutions:

A. Remove demographic features in model training: this is slightly decreasing the model performance but no significant harm.

Model Architecture	Demographic Features	Root Mean Squared Error (RMSE)	R^2
Boosted tree regressor	included	2376	0.78
Boosted tree regressor	excluded	2574	0.74

B. Customise discount rule(s) for each product based on historical sales: "discriminate" against products, not customers.
C. Use personalised advertising instead of personalised pricing/discount-offering to target predicted purchase amounts of individuals. This is giving customers the freedom to choose but will increase profit as we target the advertising based on predicted amount a customer's willing to pay.