This is task 3 of Cognizant Artificial Intelligence virtual program

The findings of this are as follows

Link to answer to this file is – https://drive.google.com/file/d/1aaTH3555PaHY9DT3iegLa5AHosbYBzSj/view?usp=sharing

Section 1 - Setup

First, we need to mount this notebook to our Google Drive folder, in order to access the CSV data file. If you haven't already, watch this video https://www.youtube.com/watch?v=woHxvbBLarQ to help you mount your Google Drive folder.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

We want to use dataframes once again to store and manipulate the data.

```
!pip install pandas
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (1.3.5)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.7/dist-packages (from pandas) (1.21.6)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas) (2.8.2)
```


Section 2 - Data loading

Similar to before, let's load our data from Google Drive for the 3 datasets provided. Be sure to upload the datasets into Google Drive, so that you can access them here.

```
path = "/content/drive/MyDrive/Forage_Cognizant/3_Model_Building/"
sales_df = pd.read_csv(f"{path}sales.csv")
sales_df.drop(columns=["Unnamed: 0"], inplace=True, errors='ignore')
sales_df.head()
```

	transaction_id	timestamp	product_id	category	customer_type	unit_price	quantity	total	payment_type
0	a1c82654-c52c-45b3-8ce8-4c2a1efe63ed	2022-03-02 09:51:38	3bc6c1ea-0198-46de-9ffd-514ae3338713	fruit	gold	3.99	2	7.98	e-wallet
1	931ad550-09e8-4da6-beaa-8c9d17be9c60	2022-03-06 10:33:59	ad81b46c-bf38-41cf-9b54-5fe7f5eba93e	fruit	standard	3.99	1	3.99	e-wallet
2	ae133534-6f61-4cd6-b6b8-d1c1d8d90aea	2022-03-04 17:20:21	7c55cbd4-f306-4c04-a030-628cbe7867c1	fruit	premium	0.19	2	0.38	e-wallet
3	157cebd9-aaf0-475d-8a11-7c8e0f5b76e4	2022-03-02 17:23:58	80da8348-1707-403f-8be7-9e6deeccc883	fruit	gold	0.19	4	0.76	e-wallet
4	a81a6cd3-5e0c-44a2-826c-aea43e46c514	2022-03-05 14:32:43	7f5e86e6-f06f-45f6-bf44-27b095c9ad1d	fruit	basic	4.49	2	8.98	debit card

Data Cleaning

Let's look for missing values

```
sales_df.info()
```

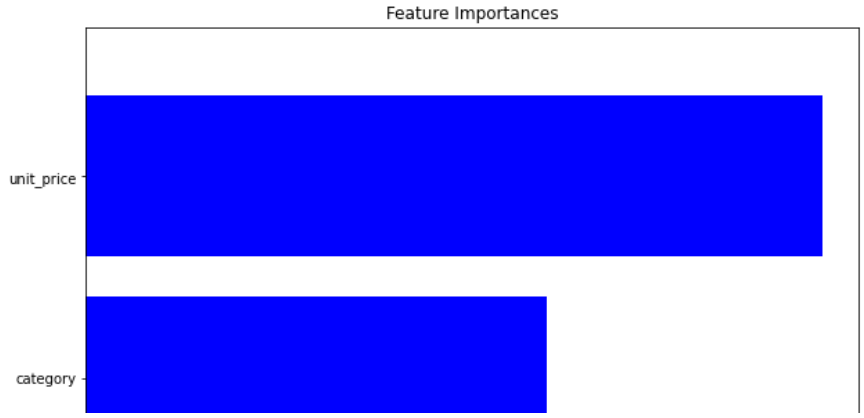
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7829 entries, 0 to 7828
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   transaction_id  7829 non-null   object
 1   timestamp       7829 non-null   object
 2   product_id      7829 non-null   object
 3   category        7829 non-null   object
 4   customer_type   7829 non-null   object
 5   unit_price      7829 non-null   float64
 6   quantity        7829 non-null   int64
```

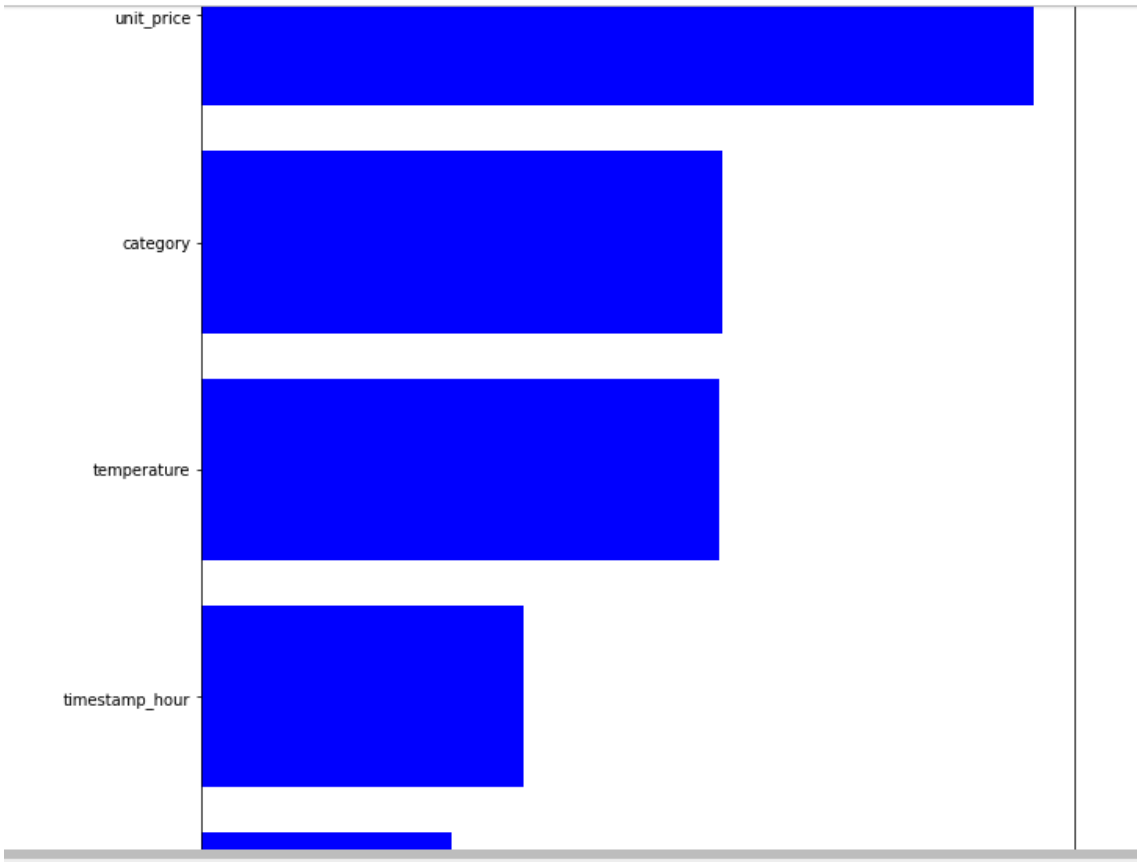
78]:
```
# from sklearn.compose import ColumnTransformer
# from sklearn.pipeline import Pipeline
# from sklearn.impute import SimpleImputer
# from sklearn.preprocessing import OrdinalEncoder

# # Preprocessing for numerical data
# numerical_transformer = SimpleImputer(strategy='constant')

# # Preprocessing for categorical data
# categorical_transformer = Pipeline(steps=[
#     ('imputer', SimpleImputer(strategy='most_frequent')),
#     ('ordinal', OrdinalEncoder())
# ])

# # Bundle preprocessing for numerical and categorical data
# preprocessor = ColumnTransformer(
#     transformers=[
#         ('num', numerical_transformer, numerical_cols),
#         ('cat', categorical_transformer, categorical_cols)
#     ])
```

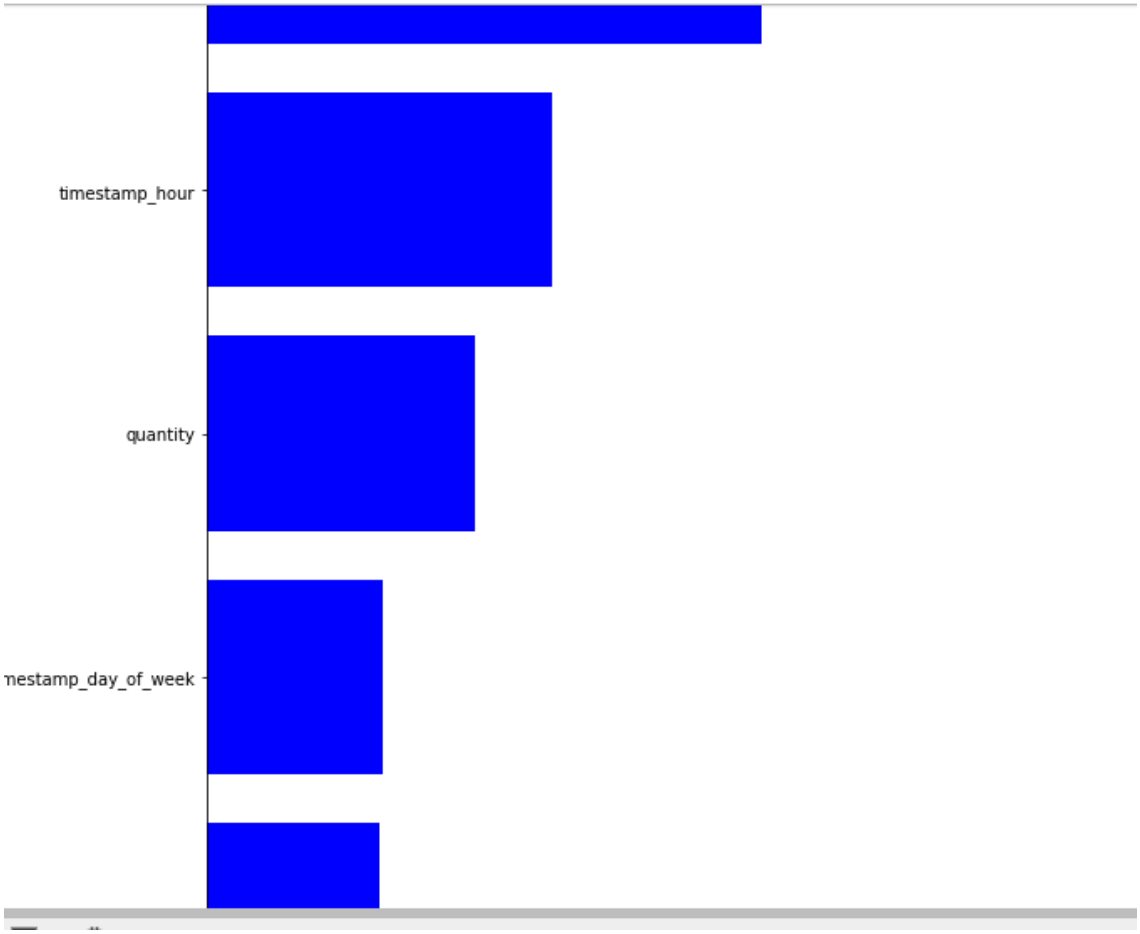
93]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import cross_val_score

model = RandomForestRegressor(n_estimators=100, random_state=0)
model.fit(X_train, y_train)
preds = model.predict(X_valid)
```

```
import numpy as np
```

194]:
```
# features = [i.split("__")[0] for i in X.columns]
features = [i for i in X.columns]
importances = model.feature_importances_
indices = np.argsort(importances)

fig, ax = plt.subplots(figsize=(10, 20))
plt.title('Feature Importances')
plt.barh(range(len(indices)), importances[indices], color='b', align='center')
plt.yticks(range(len(indices)), [features[i] for i in indices])
plt.xlabel('Relative Importance')
plt.show()
```




unit_price
category
temperature
timestamp_hour


timestamp_hour
quantity
nestamp_day_of_week