Problem statement:
Tasks:
1. In that case, what would be the business KPIs for evaluating the impact of the new ML model?
2. Which loss functions could be used for the model evaluation?
3. Imagine that the sales results are influenced by the Covid-19 spread and the inflation situation. How would you deal with this impact on the data?
4. Implement a Machine Learning model of your choice to predict the future sales of the products groups
5. What benefit would you see when using a probabilistic approach vs a single-point forecasting approach for tackling this use case?
Optional
6. Implement a Machine Learning model of your choice to estimate the uncertainty distribution and the accuracy of Conrad sales.

Answers:

1. In that case, what would be the business KPIs for evaluating the impact of the new ML model?

Many business kpis can be used for evaluating the impact of a new model such as:

- the accuracy of the model and how well it can forecast the future.
- the roi or in this particular case the profit achieved by the decision made via the model.

in the particular case of commerce applications predicting the sales can help us in inventory management or warehouse management. Particular kpis are to be used in this case:
- Inventory Turnover Ratio: this is calculated as a fraction of sales by the inventory cost. Predicting the sales enables more informed decision making in stock management. Thus, knowing in advance the sales minimizes wasted costs on inventory and warehouses which is translated by higher inventory turnover ratio and results in more profits.
- Customer satisfaction: via clients interviews and surveys it's possible to build a customer satisfaction kpi to evaluate our sales predictive analytics results. Thus, a better sales prediction leads to a better stock and inventory strategy which affect products availability and eventually customer satisfaction.
- Sales revenue: this kpi is the most useful indicator for a successful sales predictive analytics strategy.
- Cost saving: this kpi indicates we successfully anticipated the warehouse and inventory strategy and saved on stock management thanks to our model.

2. Which loss functions could be used for the model evaluation?

The sales is a continuous variable, the predicting the sales is a nonlinear regression problem and the loss functions that can be used in this case are:
- MSE: average of the squared differences between the predicted values and the actual values.
- MAE: the average absolute difference between the predicted values and the actual values.
-MAPE: average of the percentage of the differences between the predicted values and the actual values.

- RMSE: sqrt of MSE
: in the context of probalistic nn:
-negative log likelihood: logarithm of the reciprocal of the likelihood function.

- Kullback-Leibler divergence: compares how good the estimated probability distribution to the real one.

3. Imagine that the sales results are influenced by the Covid-19 spread and the inflation situation. How would you deal with this impact on the data?

The seasonality captured by the model for predicting the sales time series represents the behavior of this variable in normal days that are not impacted by special events like the covid 19 and the inflation.

The influence of these event that are caused by a change of behavior of the client in a result of difficult economic time will cause unexpected spikes and dips in the time series. We need to model these events as special holiday to capture their proper trends and seasonality and to avoid influencing the prediction in normal days.

4. Implement a Machine Learning model of your choice to predict the future sales of the products groups

- notebook : https://github.com/AsmaZgo/assignement_conrad/blob/main/ts_assignement.ipynb describes the steps to build the model.
- the file : https://github.com/AsmaZgo/assignement_conrad/blob/main/gbr_model.joblib is the selected model

*Methodology:

creating a model for the whole dataset. It resulted in an acceptable performance.

Now we want to inspect if the timeseries don't follow the same trends and similarities.

The first intuition consists in creating a model for each product group. But with this method we obtained worse result for the first test (not enough data).

The final method consists in analyzing the similarity between the timeseries, clustering them and finally producing a model for each group.

In this version, we will create only one model as a perspective we can consider creating 4 model for each product group sales time series cluster (the similarity between the timeseries analysis can be found in this notebook: https://github.com/AsmaZgo/assignement_conrad/blob/main/ts_assignement_data_analysis.ipynb)

5. What benefit would you see when using a probabilistic approach vs a single-point forecasting approach for tackling this use case?

- Probalistic approach generalizes better than single point approach
- quantify the uncertainty in their predictive output
- minimize risk since quantifying the uncertainty distribution highlights higher risks (large uncertainty intervals) or smaller ones (lower intervals)
- better decision making thanks to evaluating ranges of values instead of just one value

Optional

6. Implement a Machine Learning model of your choice to estimate the uncertainty distribution and the accuracy of Conrad sales.

The following approaches are followed for selecting the best model for predicting the uncertainty distribution of Conrad sales:

(1) Implementing a Bayesian regression model, two solutions are examined:
   - using naïve bayes first but due to the linear nature of this model it leads to very bad results
   - using a Bayesian neural network with an output layer that follows a normal distribution (as a result of the observation done in the analysis)

(2) Implementing an ensemble method. Perspective (because of the time limit we couldn't explore this option.)

Selected model training for question 4

```
from sklearn.ensemble import GradientBoostingRegressor

from sklearn.metrics import mean_absolute_error
gbr =
GradientBoostingRegressor(random_state=10,learning_rate=0.1,n_estimator
s=1000)
gbr.fit(X_train, y=y_train)
gbr_pred = gbr.predict(X_test)
```

Selected model training for question 6

```
import keras
def posterior(kernel_size, bias_size, dtype=None):
    n = kernel_size + bias_size
    posterior_model = keras.Sequential(
        [
            tfp.layers.VariableLayer(
                tfp.layers.MultivariateNormalTriL.params_size(n),
dtype=dtype
            ),
            tfp.layers.MultivariateNormalTriL(n),
        ]
    )
    return posterior_model


import tensorflow as tf
import tensorflow_probability as tfp
import pandas as pd
import numpy as np


# Convert the data to a TensorFlow dataset
dataset = tf.data.Dataset.from_tensor_slices((
    X_train[["productsGroup_key", "year", "month", "day_of_week",
"day", "quarter", "weekofyear"]].values,
    y_train.values))
```

```
# Define the number of features and the number of product groups
num_features = 7  # productsGroup_key, year, month, day_of_week, day,
quarter, weekofyear
num_groups = y_train.nunique()
print(num_groups)
# Define the model using Keras functional API
input_layer = tf.keras.Input(shape=(num_features,))
hidden_layer = tfp.layers.DenseFlipout(64,
activation="relu")(input_layer)
hidden_layer = tfp.layers.DenseFlipout(64,
activation="relu")(hidden_layer)
#output_layer = tfp.layers.DenseFlipout(1)(hidden_layer)
distribution_params = keras.layers.Dense(units=2)(hidden_layer)
output_layer = tfp.layers.IndependentNormal(1)(distribution_params)
model = tf.keras.Model(inputs=input_layer, outputs=output_layer)

# Define the loss function as the negative log likelihood
negloglik = lambda y, rv_y: -rv_y.log_prob(y)

# Define the prior over the weights
prior = tfp.distributions.Normal(loc=0., scale=1.)

# Define the posterior over the weights using the variational inference
posteriors =  posterior

# Define the loss function as the negative log likelihood
negloglik = lambda y, rv_y: -tf.reduce_mean(rv_y.log_prob(y))

# Define the optimizer and compile the model
optimizer = tf.keras.optimizers.Adam(learning_rate=0.01)
model.compile(optimizer=optimizer, loss="mse")
#model.compile(optimizer=optimizer, loss=negative_loglikelihood)

# Train the model
history = model.fit(dataset.batch(128), epochs=40, verbose=2)
```