Here is the background information on your task

Your work on the previous task was very helpful to propel this project forward with the client. Based on your recommendations, they want to focus on the following problem statement:

“Can we accurately predict the stock levels of products based on sales data and sensor data on an hourly basis in order to more intelligently procure products from our suppliers?”

The client has agreed to share more data in the form of sensor data. They use sensors to measure temperature storage facilities where products are stored in the warehouse, and they also use stock levels within the refrigerators and freezers in store.

It is your task to look at the data model diagram that has been provided by the Data Engineering team and to decide on what data you’re going to use from the data available. In addition, we need you to create a strategic plan as to how you’ll use this data to complete the work to answer the problem statement.

You can summarize your choices and plan of work in a PowerPoint presentation. This PowerPoint will be sent to the Data Science team leader and the client for a review. Make sure to keep it concise (ideally 1 slide) and business-friendly.

Here is your task

Step 1: Data modeling

Look at the data model provided in the additional resources. Look at all the data that is now available from the client and decide what you want to use for the modeling of the problem statement. This should take 10-15 minutes.

Step 2: Strategic planning

Come up with a plan as to how you’ll use this data to solve the problem statement that the client has positioned. This plan will be used to describe to the client how we are planning to complete the remaining work and to build trust with the client as a domain expert. If you need some guidance, use the provided resource video that describes the high-level overview of a data science project. This should take 10-15 minutes.

Step 3: Communication

Summarize the data that you want to make use of and the strategic plan of action in a single PowerPoint slide. This will be sent to the Data Science team leader and the client, so be sure to be concise and use business-friendly language.

Here is the background information on your task(getting started with some ml)

The client has provided 3 datasets, it is now your job to combine, transform and model these datasets in a suitable way to answer the problem statement that the business has requested.

Most importantly, once the modeling process is complete, we need you to communicate your work and analysis in the form of a single PowerPoint slide, so that we can present the results back to the business. The key here is to use business-friendly language and to explain your results in a way that the business will understand. For example, ensure that when you’re summarizing the performance of the results you don’t use technical metrics, but rather convert it into numbers that they’ll understand.

Here is your task

You have previously outlined the strategic plan for completing the modeling work based on the problem statement.

It is now time for you to deliver on this strategic plan. Some additional resources have been provided to help you with this task. If you feel comfortable enough to complete this task on your own, use the “modeling.ipynb” Python notebook to get started. However, if you’re not sure where to start and would like to be guided through the modeling process, use the “modeling\_walkthrough.ipynb” notebook.

Feel free to use these notebooks in a similar way that we did before, using Google Colab. If you prefer to use them within your own development environment, that’s fine too.

The modeling process should take 35-45 minutes.

Once you’re done with the modeling, summarize your results in a business-friendly single PowerPoint slide. Be sure to explain whether this model can help them to tackle the problem statement.

Here is the background information on your task

Gala Groceries saw the results of the machine learning model as promising and believe that with more data and time, it can add real value to the business.

To build the foundation for this machine learning use case, they want to implement a first version of the algorithm into production. In the current state, as a Python notebook, this is not suitable to productionize a machine learning model.

Therefore, as the Data Scientist that created this algorithm, it is your job to prepare a Python module that contains code to train a model and output the performance metrics when the file is run.

Move on to the next step to get started.

Here is your task

Additional information about Python modules and running Python files is provided in the additional resources. You can assume for this task that the Python file does not need to process, clean or transform the dataset. The Python file should be able to load a CSV file into a data frame, then immediately start training on that data. Assume that the CSV file will contain the same columns as the dataset that you trained the model on in the previous task.

Be sure to write good quality code, this means following best practices and writing your code in a clear and uniform manner. More information about best practices are provided in the additional resources. Furthermore, make sure to document your code with comments, as this will help the ML engineering team to understand what you’ve written.

When you’re done, submit this file for review by the ML engineering team.

Step 1: Plan

Good quality code should be planned and should follow a uniform and clear structure. Before you start writing the Python module, take some time to think about how you want to structure this file and what needs to be included from your notebook. It is your choice whether you’d rather write it all in 1 block, separate it out into functions, or create a class with methods. Depending on your ability with coding, you may take a different approach. A starter file “module\_starter.py” is provided, which gives some hints as to how you may want to structure the file. If you’re a beginner, you may also want to make use of the file named “module\_helper.py”, which includes some functions that you may want to make use of. If you do use this helper file, you can simply copy and paste the functions that you would like to use into the module that you’re writing. This should take 5-10 minutes.

Step 2: Write

After planning the module that you’re going to write, you can start creating your file! Be sure to follow a consistent and clear structure, and use the additional resources for best practices. Also be sure to include plenty of comments and documentation, because the ML engineering team is not the team that wrote this code. Remember, you can assume that the Python file does not need to process, clean or transform the dataset. You can load the Python file as a CSV file directly into a data frame, and then immediately start training on that data. Assume that the CSV file will contain the same columns as the dataset that you trained the model on in the previous task. This should take you 30-40 minutes.

# ------- BEFORE STARTING - SOME BASIC TIPS

# You can add a comment within a Python file by using a hashtag '#'

# Anything that comes after the hashtag on the same line, will be considered

# a comment and won't be executed as code by the Python interpreter.

# --- 1) IMPORTING PACKAGES

# The first thing you should always do in a Python file is to import any

# packages that you will need within the file. This should always go at the top

# of the file

# --- 2) DEFINE GLOBAL CONSTANTS

# Constants are variables that should remain the same througout the entire running

# of the module. You should define these after the imports at the top of the file.

# You should give global constants a name and ensure that they are in all upper

# case, such as: UPPER\_CASE

# --- 3) ALGORITHM CODE

# Next, we should write our code that will be executed when a model needs to be

# trained. There are many ways to structure this code and it is your choice

# how you wish to do this. The code in the 'module\_helper.py' file will break

# the code down into independent functions, which is 1 option.

# Include your algorithm code in this section below:

# --- 4) MAIN FUNCTION

# Your algorithm code should contain modular code that can be run independently.

# You may want to include a final function that ties everything together, to allow

# the entire pipeline of loading the data and training the algorithm to be run all

# at once

import pandas as pd

from sklearn.ensemble import RandomForestRegressor

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import mean\_absolute\_error

from sklearn.preprocessing import StandardScaler

# Load data

def load\_data(path: str = "/path/to/csv/"):

"""

This function takes a path string to a CSV file and loads it into

a Pandas DataFrame.

:param path (optional): str, relative path of the CSV file

:return df: pd.DataFrame

"""

df = pd.read\_csv(f"{path}")

df.drop(columns=["Unnamed: 0"], inplace=True, errors='ignore')

return df

# Create target variable and predictor variables

def create\_target\_and\_predictors(

data: pd.DataFrame = None,

target: str = "estimated\_stock\_pct"

):

"""

This function takes in a Pandas DataFrame and splits the columns

into a target column and a set of predictor variables, i.e. X & y.

These two splits of the data will be used to train a supervised

machine learning model.

:param data: pd.DataFrame, dataframe containing data for the

model

:param target: str (optional), target variable that you want to predict

:return X: pd.DataFrame

y: pd.Series

"""

# Check to see if the target variable is present in the data

if target not in data.columns:

raise Exception(f"Target: {target} is not present in the data")

X = data.drop(columns=[target])

y = data[target]

return X, y

# Train algorithm

def train\_algorithm\_with\_cross\_validation(

X: pd.DataFrame = None,

y: pd.Series = None

):

"""

This function takes the predictor and target variables and

trains a Random Forest Regressor model across K folds. Using

cross-validation, performance metrics will be output for each

fold during training.

:param X: pd.DataFrame, predictor variables

:param y: pd.Series, target variable

:return

"""

# Create a list that will store the accuracies of each fold

accuracy = []

# Enter a loop to run K folds of cross-validation

for fold in range(0, K):

# Instantiate algorithm and scaler

model = RandomForestRegressor()

scaler = StandardScaler()

# Create training and test samples

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, train\_size=SPLIT, random\_state=42)

# Scale X data, we scale the data because it helps the algorithm to converge

# and helps the algorithm to not be greedy with large values

scaler.fit(X\_train)

X\_train = scaler.transform(X\_train)

X\_test = scaler.transform(X\_test)

# Train model

trained\_model = model.fit(X\_train, y\_train)

# Generate predictions on test sample

y\_pred = trained\_model.predict(X\_test)

# Compute accuracy, using mean absolute error

mae = mean\_absolute\_error(y\_true=y\_test, y\_pred=y\_pred)

accuracy.append(mae)

print(f"Fold {fold + 1}: MAE = {mae:.3f}")

# Finish by computing the average MAE across all folds

print(f"Average MAE: {(sum(accuracy) / len(accuracy)):.2f}")

ANSWER

# ------- BEFORE STARTING - SOME BASIC TIPS

# You can add a comment within a Python file by using a hashtag '#'

# Anything that comes after the hashtag on the same line, will be considered

# a comment and won't be executed as code by the Python interpreter.

# --- 1) IMPORTING PACKAGES

# The first thing you should always do in a Python file is to import any

# packages that you will need within the file. This should always go at the top

# of the file

import pandas as pd

from sklearn.ensemble import RandomForestRegressor

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import mean\_absolute\_error

from sklearn.preprocessing import StandardScaler

# --- 2) DEFINE GLOBAL CONSTANTS

# Constants are variables that should remain the same througout the entire running

# of the module. You should define these after the imports at the top of the file.

# You should give global constants a name and ensure that they are in all upper

# case, such as: UPPER\_CASE

# K is used to define the number of folds that will be used for cross-validation

K = 10

# Split defines the % of data that will be used in the training sample

# 1 - SPLIT = the % used for testing

SPLIT = 0.75

# --- 3) ALGORITHM CODE

# Next, we should write our code that will be executed when a model needs to be

# trained. There are many ways to structure this code and it is your choice

# how you wish to do this. The code in the 'module\_helper.py' file will break

# the code down into independent functions, which is 1 option.

# Include your algorithm code in this section below:

# Load data

def load\_data(path: str = "/path/to/csv/"):

"""

This function takes a path string to a CSV file and loads it into

a Pandas DataFrame.

:param path (optional): str, relative path of the CSV file

:return df: pd.DataFrame

"""

df = pd.read\_csv(f"{path}")

df.drop(columns=["Unnamed: 0"], inplace=True, errors='ignore')

return df

# Create target variable and predictor variables

def create\_target\_and\_predictors(

data: pd.DataFrame = None,

target: str = "estimated\_stock\_pct"

):

"""

This function takes in a Pandas DataFrame and splits the columns

into a target column and a set of predictor variables, i.e. X & y.

These two splits of the data will be used to train a supervised

machine learning model.

:param data: pd.DataFrame, dataframe containing data for the

model

:param target: str (optional), target variable that you want to predict

:return X: pd.DataFrame

y: pd.Series

"""

# Check to see if the target variable is present in the data

if target not in data.columns:

raise Exception(f"Target: {target} is not present in the data")

X = data.drop(columns=[target])

y = data[target]

return X, y

# Train algorithm

def train\_algorithm\_with\_cross\_validation(

X: pd.DataFrame = None,

y: pd.Series = None

):

"""

This function takes the predictor and target variables and

trains a Random Forest Regressor model across K folds. Using

cross-validation, performance metrics will be output for each

fold during training.

:param X: pd.DataFrame, predictor variables

:param y: pd.Series, target variable

:return

"""

# Create a list that will store the accuracies of each fold

accuracy = []

# Enter a loop to run K folds of cross-validation

for fold in range(0, K):

# Instantiate algorithm and scaler

model = RandomForestRegressor()

scaler = StandardScaler()

# Create training and test samples

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, train\_size=SPLIT, random\_state=42)

# Scale X data, we scale the data because it helps the algorithm to converge

# and helps the algorithm to not be greedy with large values

scaler.fit(X\_train)

X\_train = scaler.transform(X\_train)

X\_test = scaler.transform(X\_test)

# Train model

trained\_model = model.fit(X\_train, y\_train)

# Generate predictions on test sample

y\_pred = trained\_model.predict(X\_test)

# Compute accuracy, using mean absolute error

mae = mean\_absolute\_error(y\_true=y\_test, y\_pred=y\_pred)

accuracy.append(mae)

print(f"Fold {fold + 1}: MAE = {mae:.3f}")

# Finish by computing the average MAE across all folds

print(f"Average MAE: {(sum(accuracy) / len(accuracy)):.2f}")

# --- 4) MAIN FUNCTION

# Your algorithm code should contain modular code that can be run independently.

# You may want to include a final function that ties everything together, to allow

# the entire pipeline of loading the data and training the algorithm to be run all

# at once

# Execute training pipeline

def run():

"""

This function executes the training pipeline of loading the prepared

dataset from a CSV file and training the machine learning model

:param

:return

"""

# Load the data first

df = load\_data()

# Now split the data into predictors and target variables

X, y = create\_target\_and\_predictors(data=df)

# Finally, train the machine learning model

train\_algorithm\_with\_cross\_validation(X=X, y=y)

Here is the background information on your task

The ML engineering team has taken your Python module and deployed the algorithm into production along with the DevOps, which is great!

Before it goes live, the DevOps team has been collecting some predictions from the algorithm and has provided these to the ML engineering team, who have performed some testing of the predictions against the actual results for ‘estimated\_stock\_pct’. The ML engineering team were testing the predictions vs actual results to see how well the algorithm is performing on “live” data.

After performing the tests, the ML engineering team wants to discuss with you some questions about the algorithm in order to further improve the model before the DevOps team integrates it with Gala Groceries’ live system.