DMML Assignment - Group 3

Group Member Names:

1. A Rohit Sai - 2024aa05183

2. Boddupalli Venkata Naga Lokesh - 2024aa05360

3. Lingam Vinay Naga Pavan Kumar - 2024aa05364

Drive link :
https://drive.google.com/drive/folders/1uPqLmOFKMpDCZVdodDFt3XKwSEt4fSb-?usp=sharing

1. Problem Formulation

1.1 Business Problem Statement for Customer Churn Prediction

Our business seeks to proactively identify customers at high risk of churning—those who may stop using our products or services in the near future—even though interventions could retain them. By analyzing customer data on demographics, behavior, and engagement, we aim to predict which customers are likely to leave and develop targeted retention strategies (e.g., personalized offers, improved support) to decrease churn rate, increase customer lifetime value, and maintain a profitable, loyal customer base.

1.2 Key Business Objectives

Reduce the addressable churn rate through timely interventions.

Increase customer retention and loyalty.

Boost revenue and overall business health by minimizing loss of customers.

Enable data-driven and targeted retention campaigns for at-risk customer segments.

1.3 Key data sources and their attributes

HuggingFace API - https://datasets-server.huggingface.co/rows/scikit-learn/churn-prediction

Kaggle API - https://www.kaggle.com/api/v1/datasets/download/blastchar/telco-customer-churn

1.4 Expected Outputs from the Pipeline

1. Machine learning-ready, transformed feature dataset.
2. Churn prediction model performance report (including metrics such as accuracy, recall, precision, ROC-AUC).
3. Well-organized and documented source code split into stages (ingestion, cleaning, engineering, modeling, evaluation, orchestration, reporting).

1.5 measurable evaluation metrics

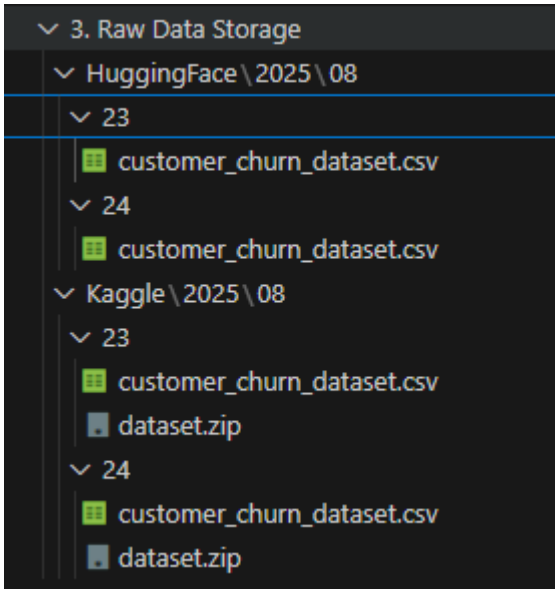
a. models are saved on each day when new training data is run and each version is saved so that we can revert anytime we want.

2. Data Ingestion

2 sources are identified – HuggingFace API and Kaggle API

Data from each source is downloaded everyday in respective folders in Raw Storage

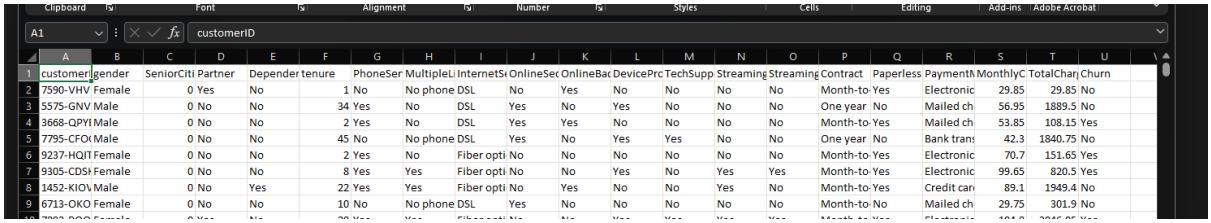
Screenshot:



Logs:

```
[2025-08-24 22:22:10] Download complete! Total rows: 5000. Saved to 3. Raw Data Storage\HuggingFace\2025\08\24\customer_chur
[2025-08-24 22:22:10] Hugging Face churn CSV saved at: 3. Raw Data Storage\HuggingFace\2025\08\24\customer_churn_dataset.csv
[2025-08-24 22:54:49] --------------------------- Kaggle ----------------------------------------
[2025-08-24 22:54:49] Starting download for dataset: blastchar/telco-customer-churn
[2025-08-24 22:54:51] Downloaded ZIP file to 3. Raw Data Storage\Kaggle\2025\08\24\dataset.zip
[2025-08-24 22:54:51] CSV extracted and saved at 3. Raw Data Storage\Kaggle\2025\08\24\customer_churn_dataset.csv
[2025-08-24 22:54:52] Kaggle churn CSV saved at: customer_churn_dataset.csv
[2025-08-24 22:54:52] --------------------------- Hugging Face ----------------------------------
[2025-08-24 22:54:52] Hugging Face churn CSV saved at:
[2025-08-24 22:55:36] --------------------------- Kaggle ----------------------------------------
[2025-08-24 22:55:36] Starting download for dataset: blastchar/telco-customer-churn
[2025-08-24 22:55:38] Downloaded ZIP file to 3. Raw Data Storage\Kaggle\2025\08\24\dataset.zip
[2025-08-24 22:55:38] CSV extracted and saved at 3. Raw Data Storage\Kaggle\2025\08\24\customer_churn_dataset.csv
[2025-08-24 22:55:38] Kaggle churn CSV saved at: customer_churn_dataset.csv
[2025-08-24 22:55:38] --------------------------- Hugging Face ----------------------------------
[2025-08-24 22:55:38] Hugging Face churn CSV saved at:
```

Data stored:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	customerI	gender	SeniorCiti	Partner	Depender	tenure	PhoneSer	MultipleLi	InternetSe	OnlineSec	OnlineBac	DevicePro	TechSupp	Streaming	Streaming	Contract	Paperless	PaymentM	MonthlyC	TotalChar	Churn
2	7590-VHV	Female	0	Yes	No	1	No	No phone	DSL	No	Yes	No	No	No	No	Month-to-	Yes	Electronic	29.85	29.85	No
3	5575-GNV	Male	0	No	No	34	Yes	No	DSL	Yes	No	Yes	No	No	No	One year	No	Mailed ch	56.95	1889.5	No
4	3668-QPY	Male	0	No	No	2	Yes	No	DSL	Yes	Yes	No	No	No	No	Month-to-	Yes	Mailed ch	53.85	108.15	Yes
5	7795-CFO	Male	0	No	No	45	No	No phone	DSL	Yes	No	Yes	Yes	No	No	One year	No	Bank trans	42.3	1840.75	No
6	9237-HQI	Female	0	No	No	2	Yes	No	Fiber opti	No	No	No	No	No	No	Month-to-	Yes	Electronic	70.7	151.65	Yes
7	9305-CDS	Female	0	No	No	8	Yes	Yes	Fiber opti	No	No	Yes	No	Yes	Yes	Month-to-	Yes	Electronic	99.65	820.5	Yes
8	1452-KIO	Male	0	No	Yes	22	Yes	Yes	Fiber opti	No	Yes	No	No	Yes	No	Month-to-	Yes	Credit car	89.1	1949.4	No
9	6713-OKO	Female	0	No	No	10	No	No phone	DSL	Yes	No	No	No	No	No	Month-to-	No	Mailed ch	29.75	301.9	No

3. **Raw Data Storage**

Earlier ingested code is stored in local server in respective folders of APIs with time stamps.

Folder / bucket Structure

HuggingFace/{year}/{month}/{date}/churn_customer_csv

Kaggle/{year}/{month}/{date}/churn_customer_csv

4. **Data Validation**

Downloaded csv's are validated against

- Duplicated rows
- Negative values
- Empty values

And respective validation_csv are generated

	A	B	C	D	E
1		missing_values	data_types	negative_values	duplicate_rows
2	Churn	2	object	-2	0
3	Contract	0	object	0	1
4	Dependents	0	object	-5	0
5	DeviceProtection	0	object	0	0
6	InternetService	4	object	0	0
7	MonthlyCharges	0	float64	0	0
8	MultipleLines	0	object	0	0
9	OnlineBackup	0	object	0	0
10	OnlineSecurity	0	object	0	0
11	PaperlessBilling	0	object	0	0
12	Partner	0	object	0	0
13	PaymentMethod	0	object	0	0
14	PhoneService	0	object	0	0
15	SeniorCitizen	0	int64	0	0
16	StreamingMovies	0	object	0	0
17	StreamingTV	0	object	0	0
18	TechSupport	0	object	0	0
19	TotalCharges	0	object	0	0

5. Data Cleaning

After the validation, same cleaned csv's are stored in different place

6. Data Transformation and Storage

In Transformation, data is prepared by encoding binary and categorical values and printing correlation values

Output of python notebook:

[ ]:
```
import matplotlib
import matplotlib.pyplot as plt
import pandas as pd
from .logger import Logger
from datetime import datetime
import os
import seaborn as sns
```

[ ]:
```
logger = Logger('../data_transformation/data_transformation_log.log')
today = datetime.today().strftime("%Y\\%m\\%d")
```

[3]:
```
cleanedDataset_path = '..\\5. Data Preparation'
csvName = 'cleaned_churn_dataset.csv'
sources = ['HuggingFace', 'Kaggle']

df_hf = pd.read_csv(os.path.join(cleanedDataset_path, sources[0], today, csvName))
df_kg = pd.read_csv(os.path.join(cleanedDataset_path, sources[1], today, csvName))

print(df_hf.shape, df_kg.shape)

master_df = pd.concat([df_hf, df_kg], ignore_index=True)
os.makedirs(os.path.join('master csv', today), exist_ok=True)
path = os.path.join('master csv', today, "cleaned_churn_dataset_master.csv")
master_df.to_csv(path, index=False)
logger.log(f'created master csv from all the date sources at {path}')
```

```
(4500, 21) (7043, 21)
```

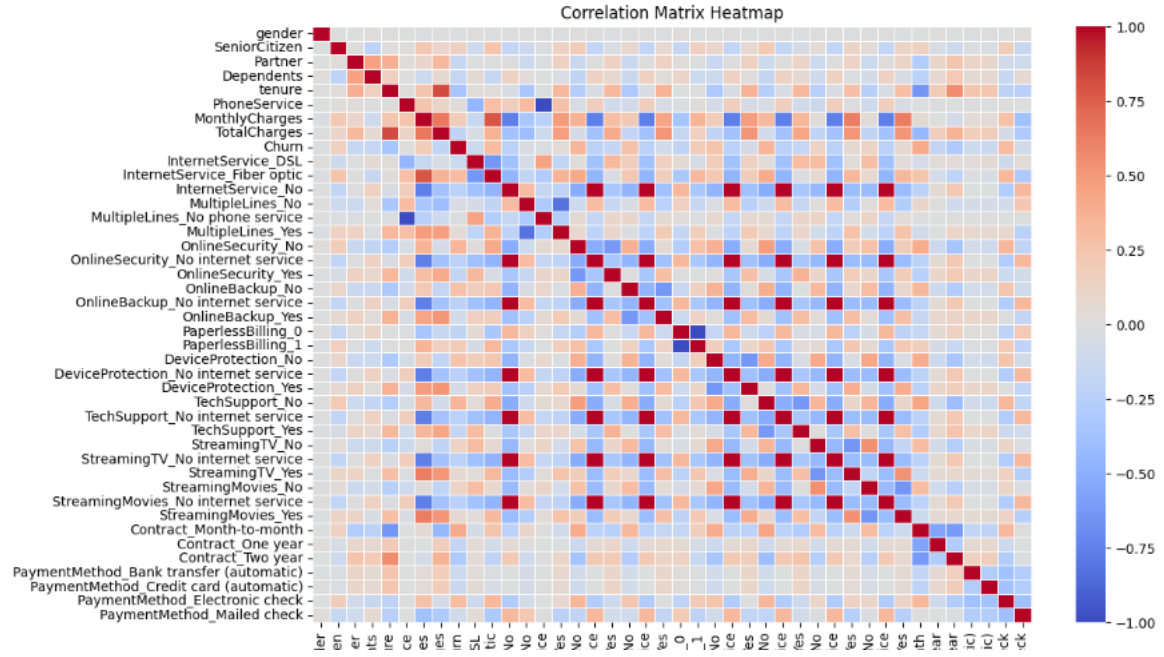
[10]:
```
df = master_df
```

[11]:
```
df["TotalCharges"] = pd.to_numeric(df["TotalCharges"], errors="coerce")

# Fill NaN with 0 (valid since tenure=0 means no charges yet)
df["TotalCharges"] = df["TotalCharges"].fillna(0)
```

[12]:
```
multiCategories = [
    "InternetService",
    "MultipleLines",
    "OnlineSecurity",
    "OnlineBackup",
    "PaperlessBilling",
    "DeviceProtection",
    "TechSupport",
    "StreamingTV",
    "StreamingMovies",
    "Contract",
```

```
# One-hot encode multi-category variables
df = pd.get_dummies(df, columns=multiCategories, drop_first=False)

bool_cols = df.select_dtypes(include=bool).columns
df[bool_cols] = df[bool_cols].astype(int)

# Compute correlation matrix
correlation_matrix = df.corr()

# Plot heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=False, cmap="coolwarm", linewidths=0.5)
plt.title("Correlation Matrix Heatmap")
plt.show()

logger.log('data transformation completed, data is ready to use for training')
```

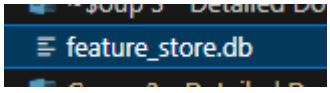
```
C:\Users\91880\AppData\Local\Temp\ipykernel_21392\3250185558.py:11: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed
in a future version. To retain the old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in to the future behavior, set `pd.set_opti
on('future.no_silent_downcasting', True)`
  df[col] = df[col].replace({"Yes": 1, "No": 0})
```

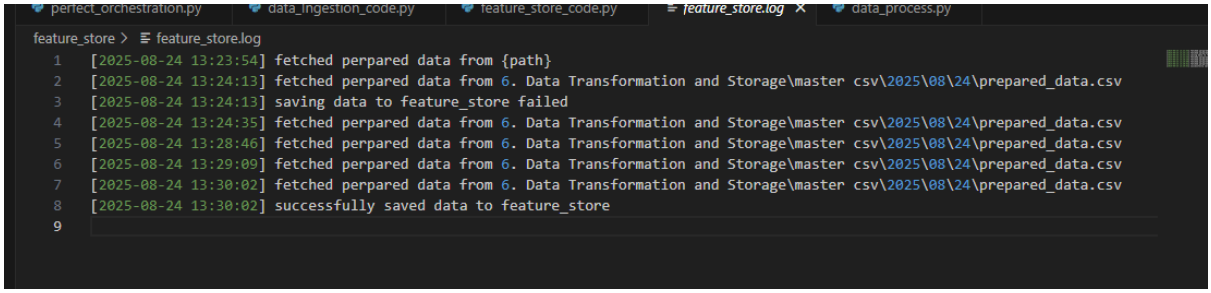


These data are stored in prepared_data.csv

7. Feature Store

Finally data in prepared_data.csv is stored in sqlite database which will be used in model training






```
path = os.path.join(preparedDataset_path, "master csv", today, csvName)
df = pd.read_csv(path)
logger.log(f"fetched perpared data from {path}")

df.columns = (
    df.columns.str.strip()       # remove leading/trailing spaces
              .str.replace(" ", "_")  # replace spaces
              .str.replace(r"[()]", "", regex=True)  # remove parentheses
)

df.to_sql("feature_store", conn, if_exists="append", index=False)
logger.log("successfully saved data to feature_store")

# Commit & close
conn.commit()
conn.close()
```

Logs are also added for every process.

8. Data Versioning

we are using github to save each and every process ingestion, validation, cleaning, transformation, feature_stores, models.

Github link:

https://github.com/vinaylingam/DMML-Assignment

with this we can revert to previous data or simple use the model from previous trainings if there are any discrepancies

9. Model Building

In this data is fetched from feature store and used in Neural networks to train model and save the versions of trained models

Code :

[1]:
```
import tensorflow as tf
from tensorflow import keras
import sqlite3
import pandas as pd
from sklearn.model_selection import train_test_split
from datetime import datetime
import os
from logger import Logger
```

[4]:
```
logger = Logger('../9. Model Building/model_building.log')
today = datetime.today().strftime("%Y\\%m\\%d")
```

[5]:
```
logger.log(f'Building Neural network model for churn dataset on {today}')
```

[6]:
```
# Connect to SQLite and load table
conn = sqlite3.connect("../feature_store.db")
try:
    df = pd.read_sql("SELECT * FROM feature_store;", conn)
    logger.log('fetched data from feature_store.db')
except:
    logger.log('failed to read data from feature_store')
conn.close()
```

[7]:
```
# Drop ID column (not useful for training)
X = df.drop(columns=["customerID", "Churn"])
y = df["Churn"]
```

[8]:
```
# Convert X and y to numpy
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Build NN
model = keras.Sequential([
    keras.layers.Dense(64, activation="relu", input_shape=(X_train.shape[1],)),
    keras.layers.Dense(32, activation="relu"),
    keras.layers.Dense(1, activation="sigmoid")  # binary classification
])

model.compile(optimizer="adam", loss="binary_crossentropy", metrics=["accuracy"])
```

```
C:\Users\91880\AppData\Roaming\Python\Python313\site-packages\keras\src\layers\core\dense.py:92: UserWarning: Do not pass an `input_shape`/`input_dim`
argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
C:\Users\91880\AppData\Roaming\Python\Python313\site-packages\keras\src\layers\core\dense.py:92: UserWarning: Do not pass an `input_shape`/`input_dim`
argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

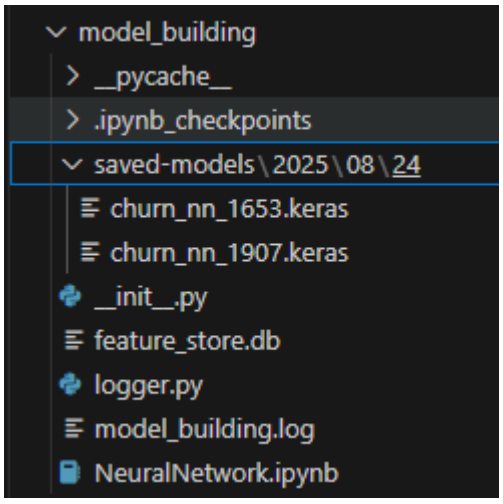
[10]:
```python
# Train
model.fit(X_train, y_train, epochs=10, batch_size=32, validation_split=0.2, verbose=1)
logger.log('model training completed')

# Evaluate
loss, acc = model.evaluate(X_test, y_test)
logger.log(f'models test accuracy: {acc}')
print("Neural Network Accuracy:", acc)
```

```
Epoch 1/10
231/231 ━━━━━━━━━━━━━━━━━━━━ 4s 5ms/step - accuracy: 0.7015 - loss: 5.1546 - val_accuracy: 0.7764 - val_loss: 0.9066
Epoch 2/10
231/231 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.7309 - loss: 2.2887 - val_accuracy: 0.8078 - val_loss: 1.3687
Epoch 3/10
231/231 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7329 - loss: 2.0556 - val_accuracy: 0.8018 - val_loss: 1.5622
Epoch 4/10
231/231 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7428 - loss: 2.3838 - val_accuracy: 0.5696 - val_loss: 1.7189
Epoch 5/10
231/231 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7417 - loss: 1.7426 - val_accuracy: 0.7493 - val_loss: 0.6680
Epoch 6/10
231/231 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7460 - loss: 1.4025 - val_accuracy: 0.6215 - val_loss: 1.2229
Epoch 7/10
231/231 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7483 - loss: 1.5984 - val_accuracy: 0.7991 - val_loss: 2.4314
Epoch 8/10
231/231 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.7552 - loss: 1.0784 - val_accuracy: 0.7753 - val_loss: 6.4822
Epoch 9/10
231/231 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7458 - loss: 1.7905 - val_accuracy: 0.8116 - val_loss: 0.8424
```

[11]:
```python
today = datetime.today().strftime("%Y\\%m\\%d")
timestamp = datetime.today().strftime("%H%M")

# make directory path only
dir_path = fr"saved-models\{today}"
os.makedirs(dir_path, exist_ok=True)

# file path inside that directory
file_path = fr"{dir_path}\churn_nn_{timestamp}.keras"

try:
    model.save(file_path)
    print(f"Model saved at: {file_path}")
    logger.log(f'model succesfully saved at {file_path}')
except:
    logger.log('failed to save the model')
```

```
Model saved at: saved-models\2025\08\24\churn_nn_1907.keras
```

Models are saved with date and time stamps



10. Orchaestration

For orchaestration, we used prefect we setup flows and tasks

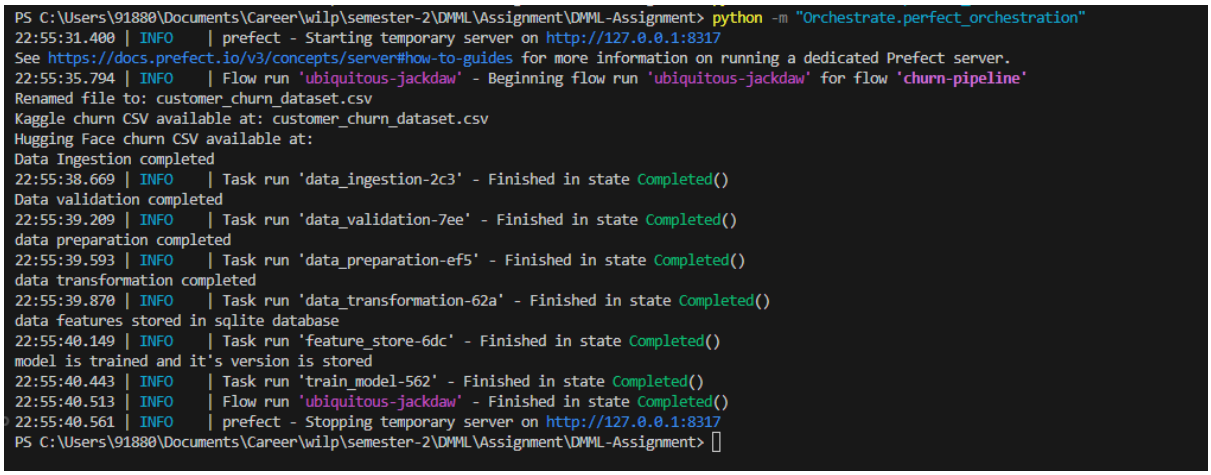
And a cron job will run the pipeline everyday at 06:00 hrs

Code:

```python
from prefect import flow, task
import papermill as pm
from data_ingestion.data_Ingestion_code import fetchData
from data_validation.data_validation_code import validateData
from data_preparation.data_process import data_prepare
from feature_store.feature_store_code import saveData

@task
def data_ingestion():
    fetchData()
    print("Data Ingestion completed")

@task
def data_validation():
    validateData()
    print("Data validation completed")

@task
def data_preparation():
    data_prepare()
    print("data preparation completed")

@task
def data_transformation():
    pm.execute_notebook(
        "data_transformation/data_transformation_code.ipynb",   # input notebook
        "data_transformation/data_transformation_code_output.ipynb",  # output notebook with executed cells
    )
    print("data transformation completed")

@task
def feature_store():
    saveData()
    print("data features stored in sqlite database")
    pass
```

```python
@task
def train_model():
    pm.execute_notebook(
        "model_building/NeuralNetwork.ipynb",   # input notebook
        "model_building/NeuralNetwork_Output.ipynb",  # output notebook with executed cells
    )
    print("model is trained and it's version is stored")
    pass

@flow
def churn_pipeline():
    data_ingestion()
    data_validation()
    data_preparation()
    data_transformation()
    feature_store()
    train_model()

if __name__ == "__main__":
    churn_pipeline()
```

Output:


PS C:\Users\91880\Documents\Career\wilp\semester-2\DMML\Assignment\DMML-Assignment> python -m "Orchestrate.perfect_orchestration"
22:55:31.400 | INFO | prefect - Starting temporary server on http://127.0.0.1:8317
See https://docs.prefect.io/v3/concepts/server#how-to-guides for more information on running a dedicated Prefect server.
22:55:35.794 | INFO | Flow run 'ubiquitous-jackdaw' - Beginning flow run 'ubiquitous-jackdaw' for flow 'churn-pipeline'
Renamed file to: customer_churn_dataset.csv
Kaggle churn CSV available at: customer_churn_dataset.csv
Hugging Face churn CSV available at:
Data Ingestion completed
22:55:38.669 | INFO | Task run 'data_ingestion-2c3' - Finished in state Completed()
Data validation completed
22:55:39.209 | INFO | Task run 'data_validation-7ee' - Finished in state Completed()
data preparation completed
22:55:39.593 | INFO | Task run 'data_preparation-ef5' - Finished in state Completed()
data transformation completed
22:55:39.870 | INFO | Task run 'data_transformation-62a' - Finished in state Completed()
data features stored in sqlite database
22:55:40.149 | INFO | Task run 'feature_store-6dc' - Finished in state Completed()
model is trained and it's version is stored
22:55:40.443 | INFO | Task run 'train_model-562' - Finished in state Completed()
22:55:40.513 | INFO | Flow run 'ubiquitous-jackdaw' - Finished in state Completed()
22:55:40.561 | INFO | prefect - Stopping temporary server on http://127.0.0.1:8317
PS C:\Users\91880\Documents\Career\wilp\semester-2\DMML\Assignment\DMML-Assignment>