Module 4 Quiz (Mayur Brijwani)

Q1. Notebook

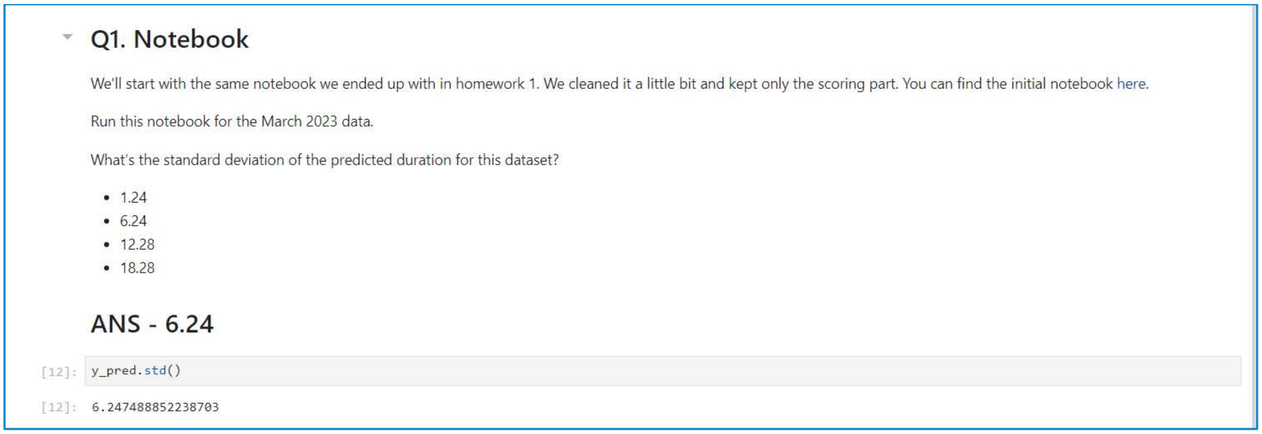
We'll start with the same notebook we ended up with in homework 1. We cleaned it a little bit and kept only the scoring part. You can find the initial notebook here.

Run this notebook for the March 2023 data.

What's the standard deviation of the predicted duration for this dataset?

- 1.24
- 6.24
- 12.28
- 18.28

ANS - 6.24



Q1. Notebook

We'll start with the same notebook we ended up with in homework 1. We cleaned it a little bit and kept only the scoring part. You can find the initial notebook here.

Run this notebook for the March 2023 data.

What's the standard deviation of the predicted duration for this dataset?

- 1.24
- 6.24
- 12.28
- 18.28

ANS - 6.24

```
[12]: y_pred.std()
```

```
[12]: 6.247488852238703
```

Q2. Preparing the output

Like in the course videos, we want to prepare the dataframe with the output.

First, let's create an artificial `ride_id` column:

df['ride_id'] = f'{year:04d}/{month:02d}_' + df.index.astype('str')

Next, write the ride id and the predictions to a dataframe with results.

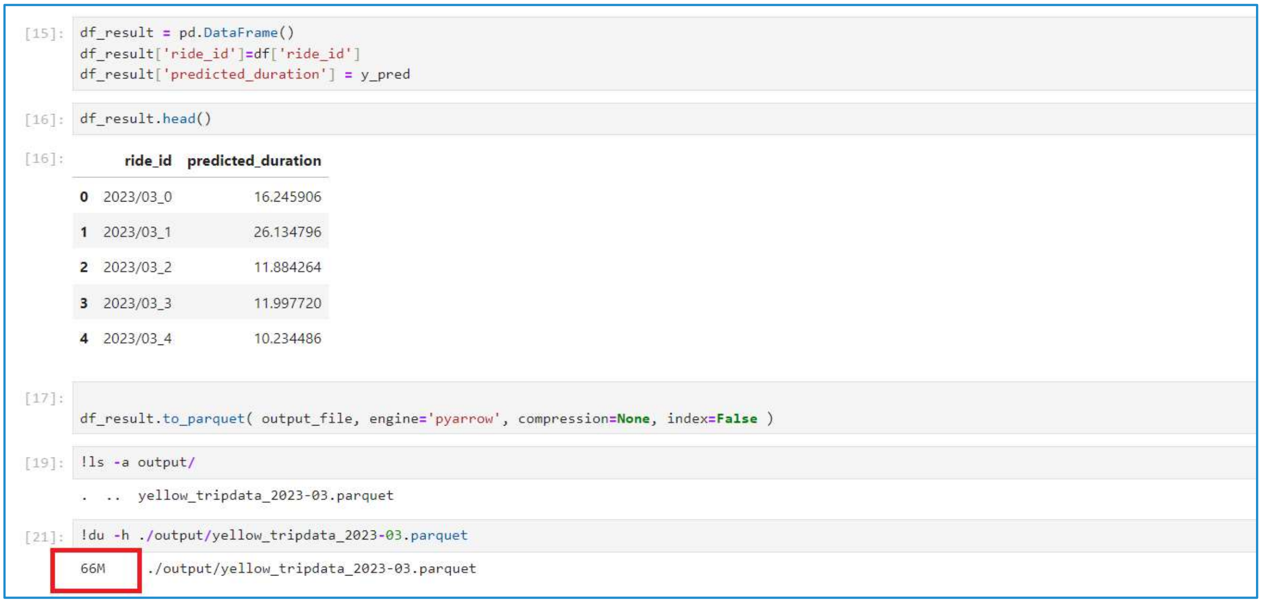
Save it as parquet:

df_result.to_parquet(output_file, engine='pyarrow', compression=None, index=False)

What's the size of the output file?

- 36M
- 46M
- 56M
- 66M

ANS - 66M



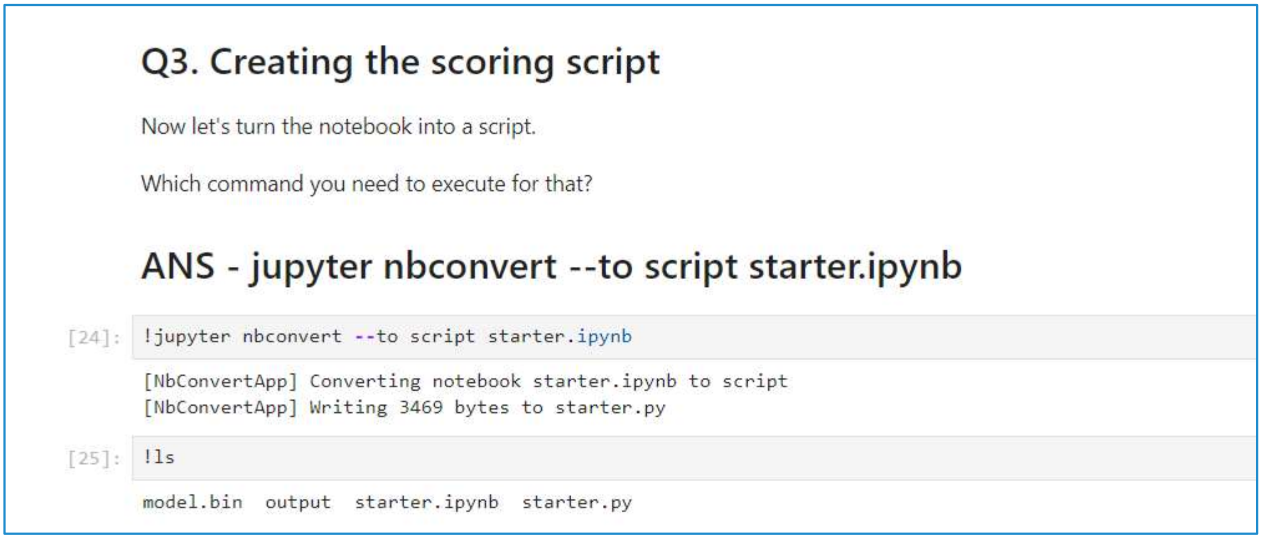
```
[15]: df_result = pd.DataFrame()
      df_result['ride_id']=df['ride_id']
      df_result['predicted_duration'] = y_pred

[16]: df_result.head()

[16]:      ride_id  predicted_duration
      0  2023/03_0           16.245906
      1  2023/03_1           26.134796
      2  2023/03_2           11.884264
      3  2023/03_3           11.997720
      4  2023/03_4           10.234486

[17]:
      df_result.to_parquet( output_file, engine='pyarrow', compression=None, index=False )

[19]: !ls -a output/
      .  ..  yellow_tripdata_2023-03.parquet

[21]: !du -h ./output/yellow_tripdata_2023-03.parquet
      66M     ./output/yellow_tripdata_2023-03.parquet
```

Q3. Creating the scoring script

Now let's turn the notebook into a script.

Which command you need to execute for that?

ANS - jupyter nbconvert --to script starter.ipynb



Q3. Creating the scoring script

Now let's turn the notebook into a script.

Which command you need to execute for that?

ANS - jupyter nbconvert --to script starter.ipynb

```
[24]: !jupyter nbconvert --to script starter.ipynb
```

```
[NbConvertApp] Converting notebook starter.ipynb to script
[NbConvertApp] Writing 3469 bytes to starter.py
```

```
[25]: !ls
```

```
model.bin  output  starter.ipynb  starter.py
```

Q4. Virtual environment

Now let's put everything into a virtual environment. We'll use pipenv for that.

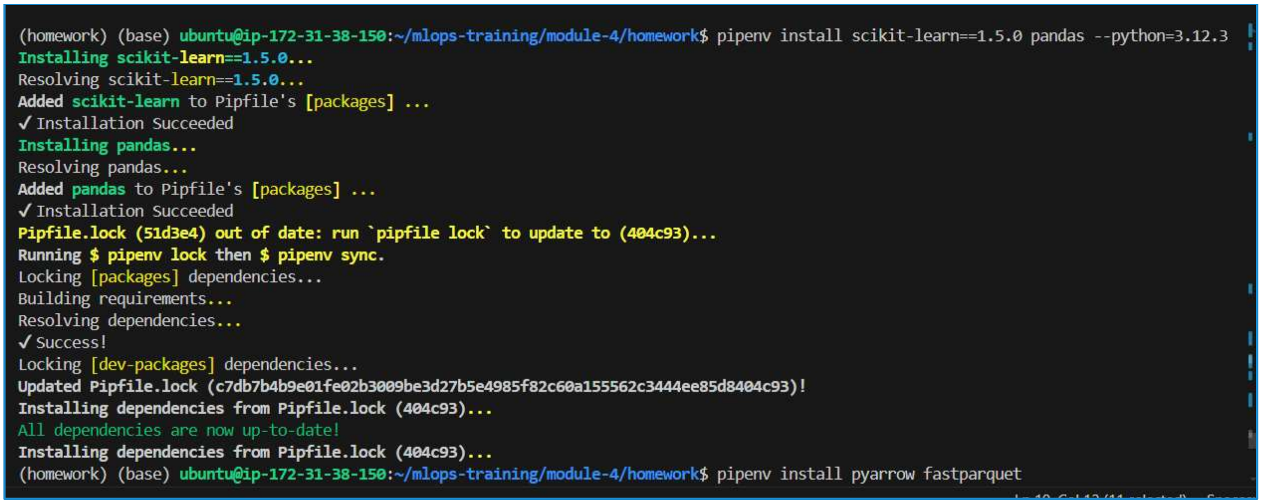
Install all the required libraries. Pay attention to the Scikit-Learn version: it should be the same as in the starter notebook.

After installing the libraries, pipenv creates two files: `Pipfile` and `Pipfile.lock`. The `Pipfile.lock` file keeps the hashes of the dependencies we use for the virtual env.

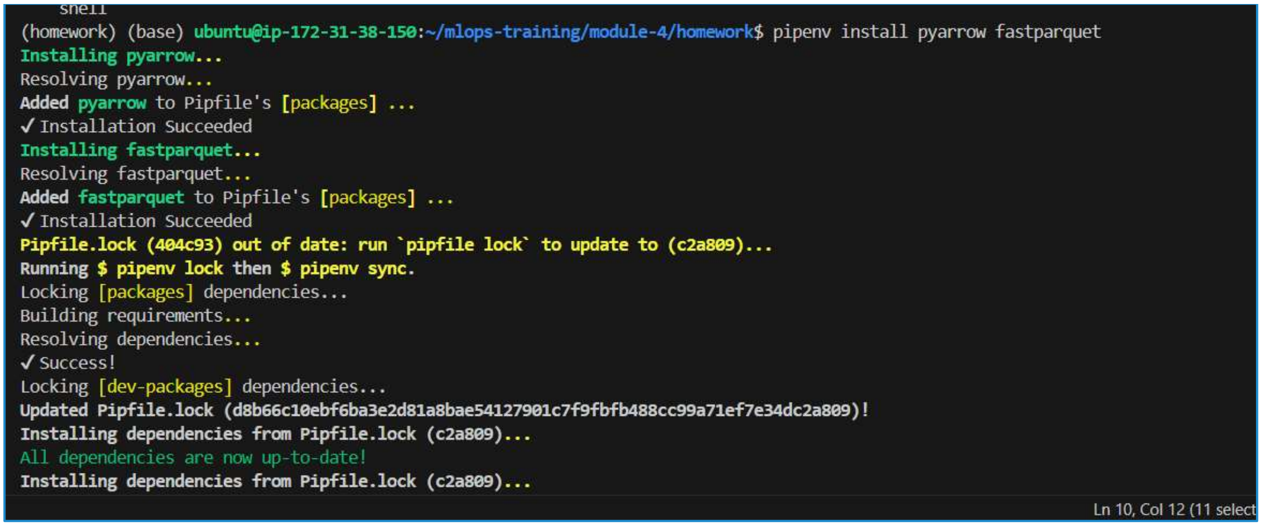
What's the first hash for the Scikit-Learn dependency?

ANS - sha256:057b991ac64b3e75c9c04b5f9395eaf19a6179244c089afdebaad98264bff37c

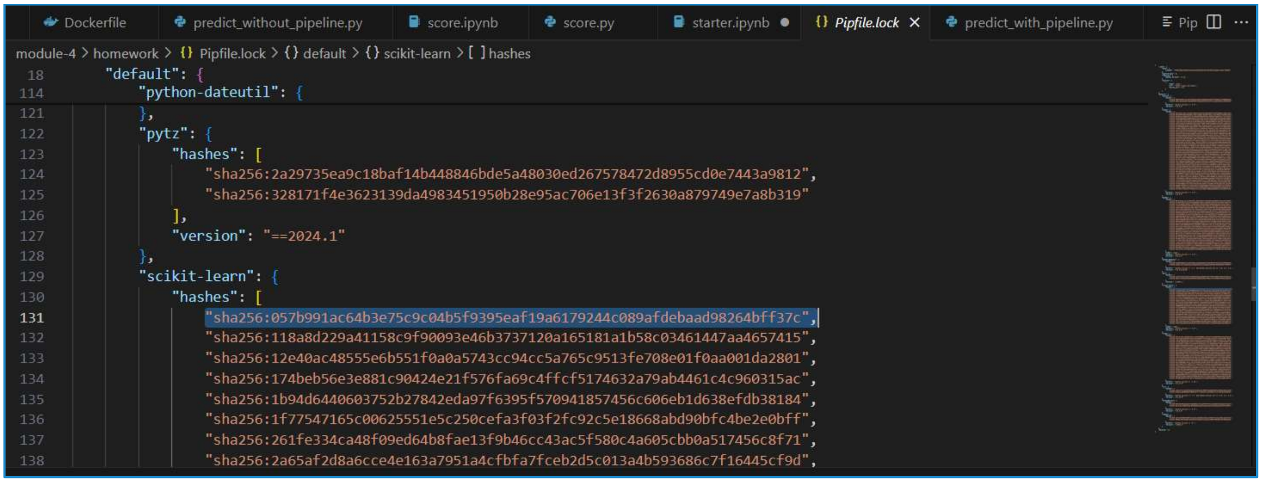
- Installing pipenv environment



```
(homework) (base) ubuntu@ip-172-31-38-150:~/mlops-training/module-4/homework$ pipenv install scikit-learn==1.5.0 pandas --python=3.12.3
Installing scikit-learn==1.5.0...
Resolving scikit-learn==1.5.0...
Added scikit-learn to Pipfile's [packages] ...
✓ Installation Succeeded
Installing pandas...
Resolving pandas...
Added pandas to Pipfile's [packages] ...
✓ Installation Succeeded
Pipfile.lock (51d3e4) out of date: run `pipfile lock` to update to (404c93)...
Running $ pipenv lock then $ pipenv sync.
Locking [packages] dependencies...
Building requirements...
Resolving dependencies...
✓ Success!
Locking [dev-packages] dependencies...
Updated Pipfile.lock (c7db7b4b9e01fe02b3009be3d27b5e4985f82c60a155562c3444ee85d8404c93)!
Installing dependencies from Pipfile.lock (404c93)...
All dependencies are now up-to-date!
Installing dependencies from Pipfile.lock (404c93)...
(homework) (base) ubuntu@ip-172-31-38-150:~/mlops-training/module-4/homework$ pipenv install pyarrow fastparquet
```



```
    shell
(homework) (base) ubuntu@ip-172-31-38-150:~/mlops-training/module-4/homework$ pipenv install pyarrow fastparquet
Installing pyarrow...
Resolving pyarrow...
Added pyarrow to Pipfile's [packages] ...
✓ Installation Succeeded
Installing fastparquet...
Resolving fastparquet...
Added fastparquet to Pipfile's [packages] ...
✓ Installation Succeeded
Pipfile.lock (404c93) out of date: run `pipfile lock` to update to (c2a809)...
Running $ pipenv lock then $ pipenv sync.
Locking [packages] dependencies...
Building requirements...
Resolving dependencies...
✓ Success!
Locking [dev-packages] dependencies...
Updated Pipfile.lock (d8b66c10ebf6ba3e2d81a8bae54127901c7f9fbfb488cc99a71ef7e34dc2a809)!
Installing dependencies from Pipfile.lock (c2a809)...
All dependencies are now up-to-date!
Installing dependencies from Pipfile.lock (c2a809)...
```


Dockerfile
predict_without_pipeline.py
score.ipynb
score.py
starter.ipynb
Pipfile.lock
predict_with_pipeline.py
Pip
module-4 > homework > {} Pipfile.lock > {} default > {} scikit-learn > [ ] hashes

```
    "default": {
        "python-dateutil": {
        },
        "pytz": {
            "hashes": [
                "sha256:2a29735ea9c18baf14b448846bde5a48030ed267578472d8955cd0e7443a9812",
                "sha256:328171f4e3623139da4983451950b28e95ac706e13f3f2630a879749e7a8b319"
            ],
            "version": "==2024.1"
        },
        "scikit-learn": {
            "hashes": [
                "sha256:057b991ac64b3e75c9c04b5f9395eaf19a6179244c089afdebaad98264bff37c",
                "sha256:118a8d229a41158c9f90093e46b3737120a165181a1b58c03461447aa4657415",
                "sha256:12e40ac48555e6b551f0a0a5743cc94cc5a765c9513fe708e01f0aa001da2801",
                "sha256:174beb56e3e881c90424e21f576fa69c4ffcf5174632a79ab4461c4c960315ac",
                "sha256:1b94d6440603752b27842eda97f6395f570941857456c606eb1d638efdb38184",
                "sha256:1f77547165c00625551e5c250cefa3f03f2fc92c5e18668abd90bfc4be2e0bff",
                "sha256:261fe334ca48f09ed64b8fae13f9b46cc43ac5f580c4a605cbb0a517456c8f71",
                "sha256:2a65af2d8a6cce4e163a7951a4cfbfa7fceb2d5c013a4b593686c7f16445cf9d",
```

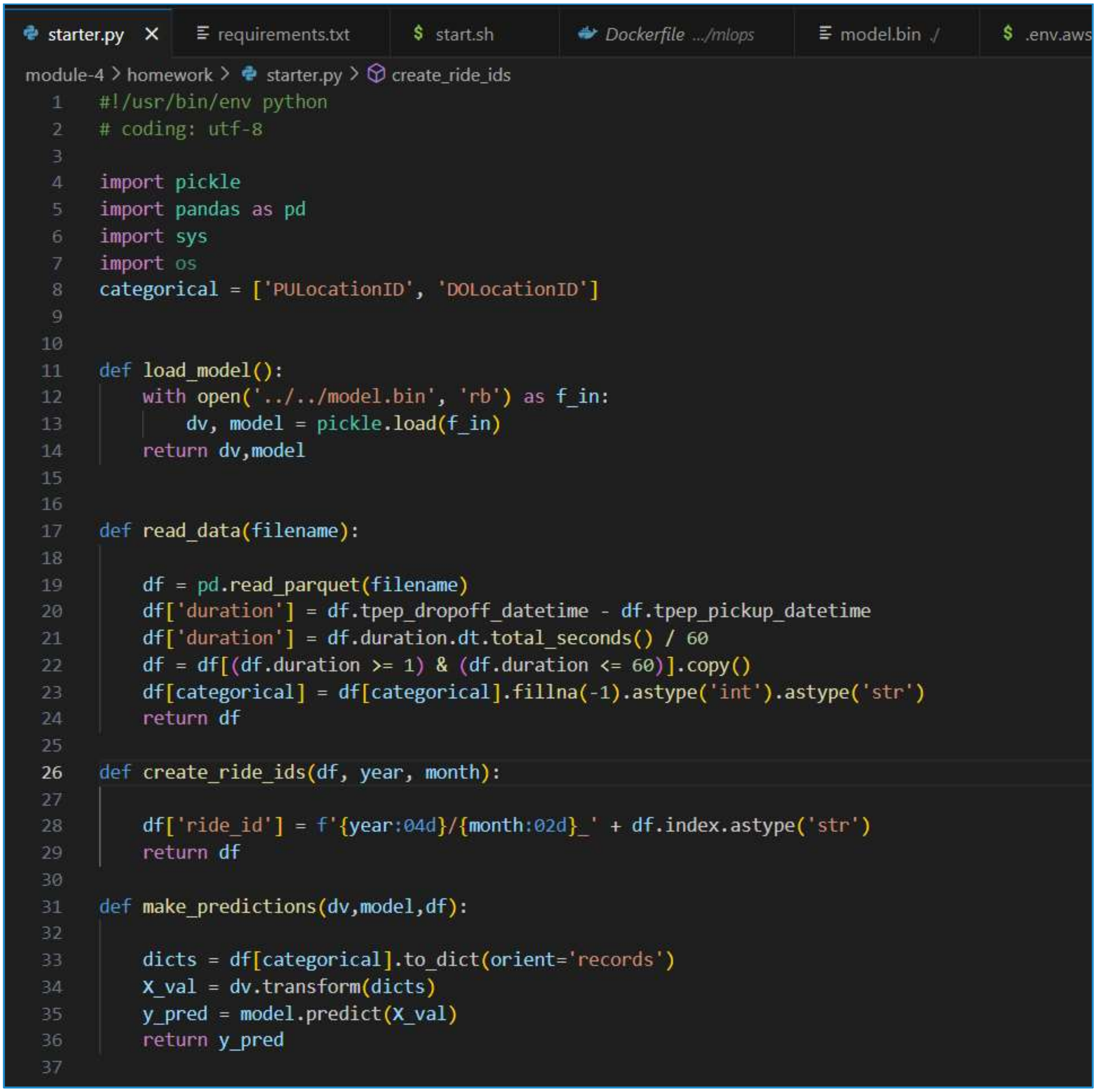
Q5. Parametrize the script

Let's now make the script configurable via CLI. We'll create two parameters: year and month.
Run the script for April 2023.
What's the mean predicted duration?

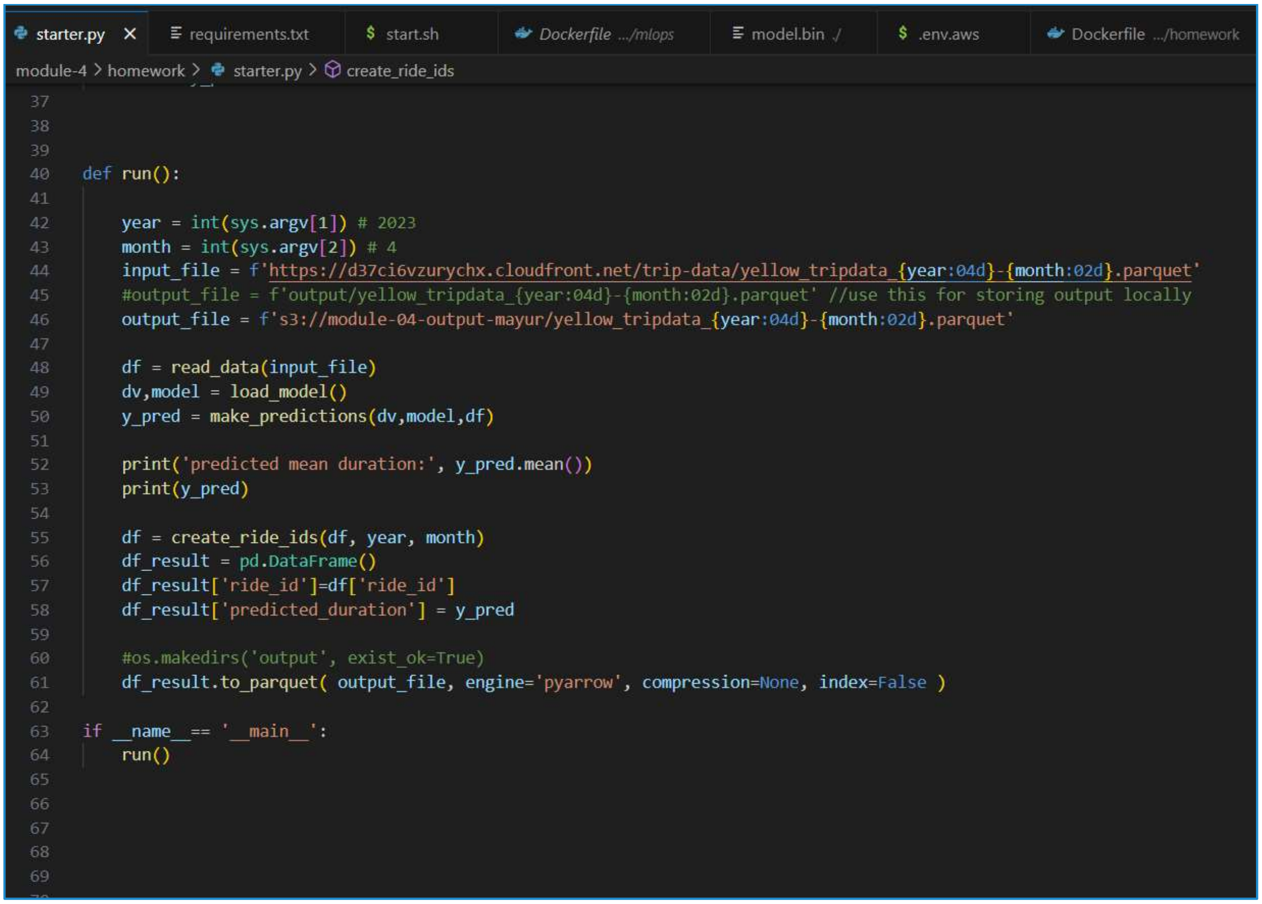
- 7.29
- 14.29
- 21.29
- 28.29

Hint: just add a print statement to your script.

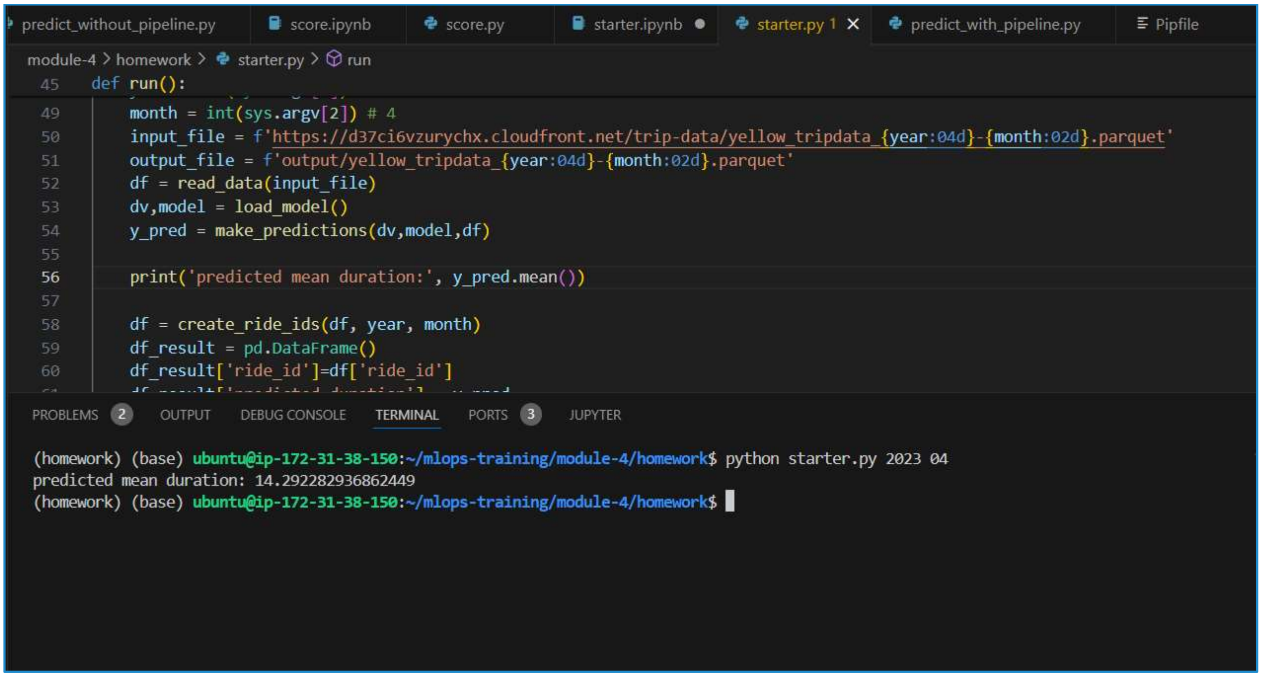
ANS: 14.29



```python
#!/usr/bin/env python
# coding: utf-8

import pickle
import pandas as pd
import sys
import os
categorical = ['PULocationID', 'DOLocationID']


def load_model():
    with open('../../model.bin', 'rb') as f_in:
        dv, model = pickle.load(f_in)
    return dv,model


def read_data(filename):

    df = pd.read_parquet(filename)
    df['duration'] = df.tpep_dropoff_datetime - df.tpep_pickup_datetime
    df['duration'] = df.duration.dt.total_seconds() / 60
    df = df[(df.duration >= 1) & (df.duration <= 60)].copy()
    df[categorical] = df[categorical].fillna(-1).astype('int').astype('str')
    return df

def create_ride_ids(df, year, month):

    df['ride_id'] = f'{year:04d}/{month:02d}_' + df.index.astype('str')
    return df

def make_predictions(dv,model,df):

    dicts = df[categorical].to_dict(orient='records')
    X_val = dv.transform(dicts)
    y_pred = model.predict(X_val)
    return y_pred

```



```python
def run():

    year = int(sys.argv[1]) # 2023
    month = int(sys.argv[2]) # 4
    input_file = f'https://d37ci6vzurychx.cloudfront.net/trip-data/yellow_tripdata_{year:04d}-{month:02d}.parquet'
    #output_file = f'output/yellow_tripdata_{year:04d}-{month:02d}.parquet' //use this for storing output locally
    output_file = f's3://module-04-output-mayur/yellow_tripdata_{year:04d}-{month:02d}.parquet'

    df = read_data(input_file)
    dv,model = load_model()
    y_pred = make_predictions(dv,model,df)

    print('predicted mean duration:', y_pred.mean())
    print(y_pred)

    df = create_ride_ids(df, year, month)
    df_result = pd.DataFrame()
    df_result['ride_id']=df['ride_id']
    df_result['predicted_duration'] = y_pred

    #os.makedirs('output', exist_ok=True)
    df_result.to_parquet( output_file, engine='pyarrow', compression=None, index=False )

if __name__== '__main__':
    run()
```



```python
def run():
    month = int(sys.argv[2]) # 4
    input_file = f'https://d37ci6vzurychx.cloudfront.net/trip-data/yellow_tripdata_{year:04d}-{month:02d}.parquet'
    output_file = f'output/yellow_tripdata_{year:04d}-{month:02d}.parquet'
    df = read_data(input_file)
    dv,model = load_model()
    y_pred = make_predictions(dv,model,df)

    print('predicted mean duration:', y_pred.mean())

    df = create_ride_ids(df, year, month)
    df_result = pd.DataFrame()
    df_result['ride_id']=df['ride_id']
```

```
(homework) (base) ubuntu@ip-172-31-38-150:~/mlops-training/module-4/homework$ python starter.py 2023 04
predicted mean duration: 14.292282936862449
(homework) (base) ubuntu@ip-172-31-38-150:~/mlops-training/module-4/homework$ 
```

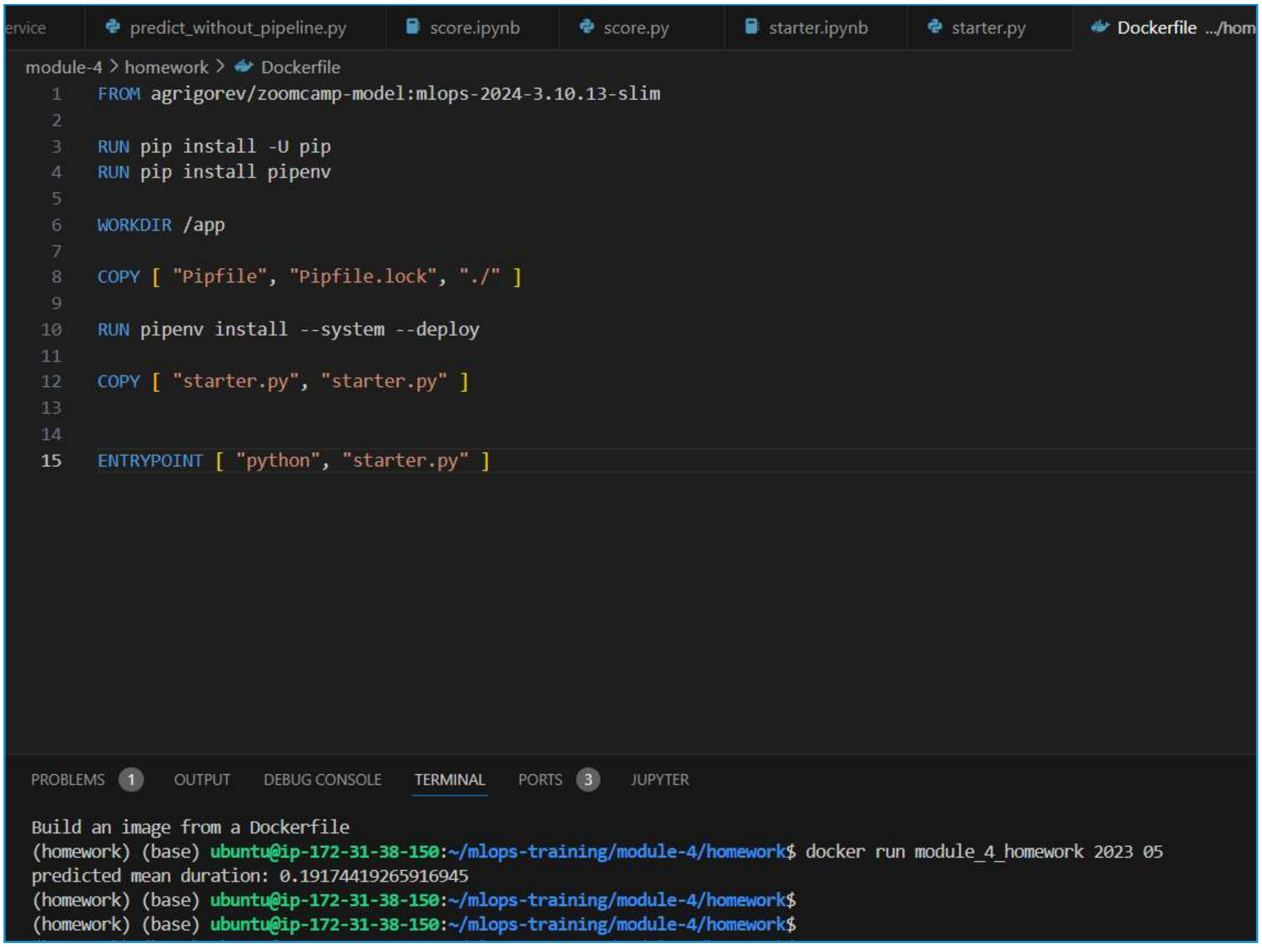
Q6. Docker container

Finally, we'll package the script in the docker container. For that, you'll need to use a base image that we prepared

Now run the script with docker. What's the mean predicted duration for May 2023?

- 0.19
- 7.24
- 14.24
- 21.19

ANS : 0.19



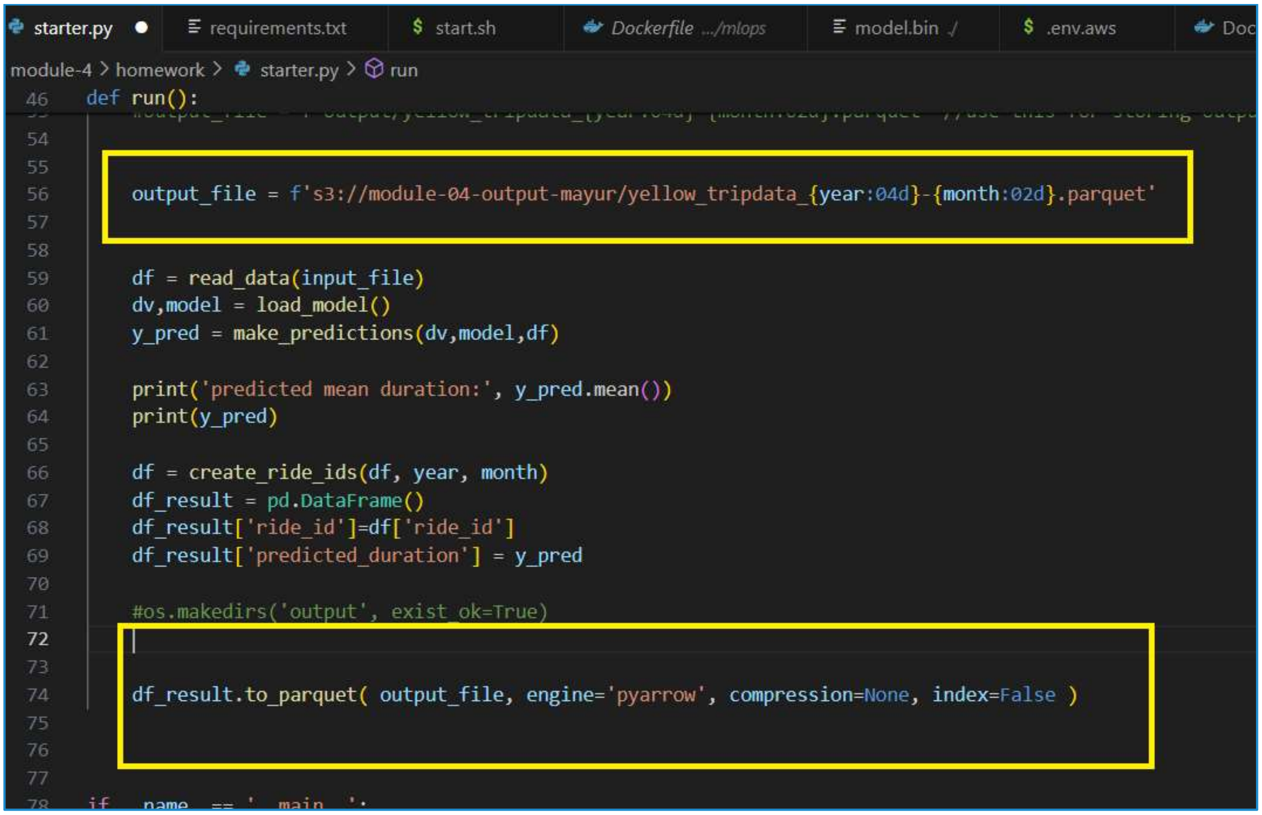
```dockerfile
FROM agrigorev/zoomcamp-model:mlops-2024-3.10.13-slim

RUN pip install -U pip
RUN pip install pipenv

WORKDIR /app

COPY [ "Pipfile", "Pipfile.lock", "./" ]

RUN pipenv install --system --deploy

COPY [ "starter.py", "starter.py" ]


ENTRYPOINT [ "python", "starter.py" ]
```

```
Build an image from a Dockerfile
(homework) (base) ubuntu@ip-172-31-38-150:~/mlops-training/module-4/homework$ docker run module_4_homework 2023 05
predicted mean duration: 0.19174419265916945
(homework) (base) ubuntu@ip-172-31-38-150:~/mlops-training/module-4/homework$
(homework) (base) ubuntu@ip-172-31-38-150:~/mlops-training/module-4/homework$
```

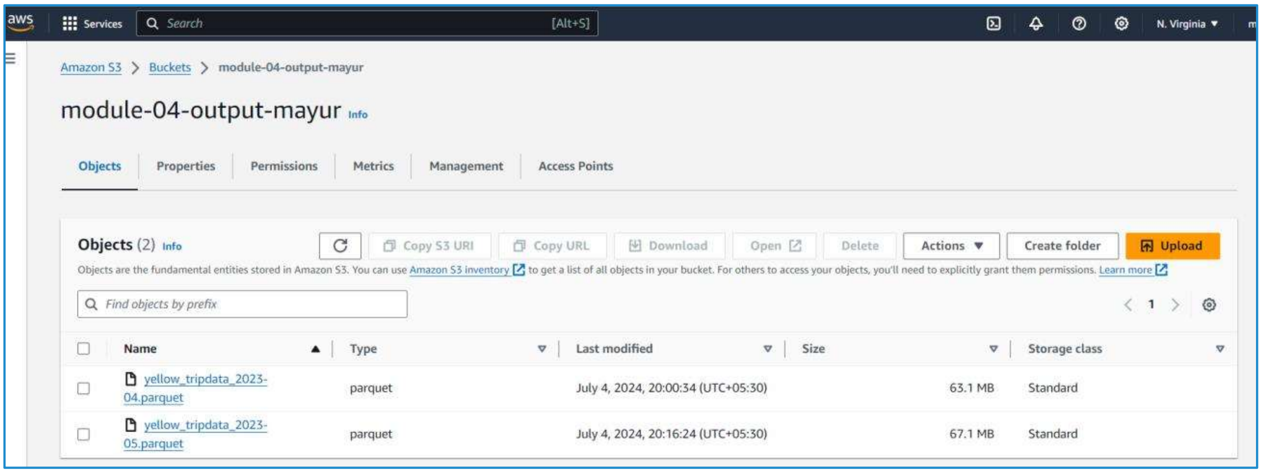
Bonus: upload the result to the cloud (Not graded)

Just printing the mean duration inside the docker image doesn't seem very practical. Typically, after creating the output file, we upload it to the cloud storage.

Modify your code to upload the parquet file to S3/GCS/etc.



```python
def run():
    #output_file = f'output/yellow_tripdata_{year:04d}-{month:02d}.parquet' //use this for storing outpu

    output_file = f's3://module-04-output-mayur/yellow_tripdata_{year:04d}-{month:02d}.parquet'

    df = read_data(input_file)
    dv,model = load_model()
    y_pred = make_predictions(dv,model,df)

    print('predicted mean duration:', y_pred.mean())
    print(y_pred)

    df = create_ride_ids(df, year, month)
    df_result = pd.DataFrame()
    df_result['ride_id']=df['ride_id']
    df_result['predicted_duration'] = y_pred

    #os.makedirs('output', exist_ok=True)


    df_result.to_parquet( output_file, engine='pyarrow', compression=None, index=False )



if __name__ == '__main__':
```



module-04-output-mayur Info

Objects (2) Info

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

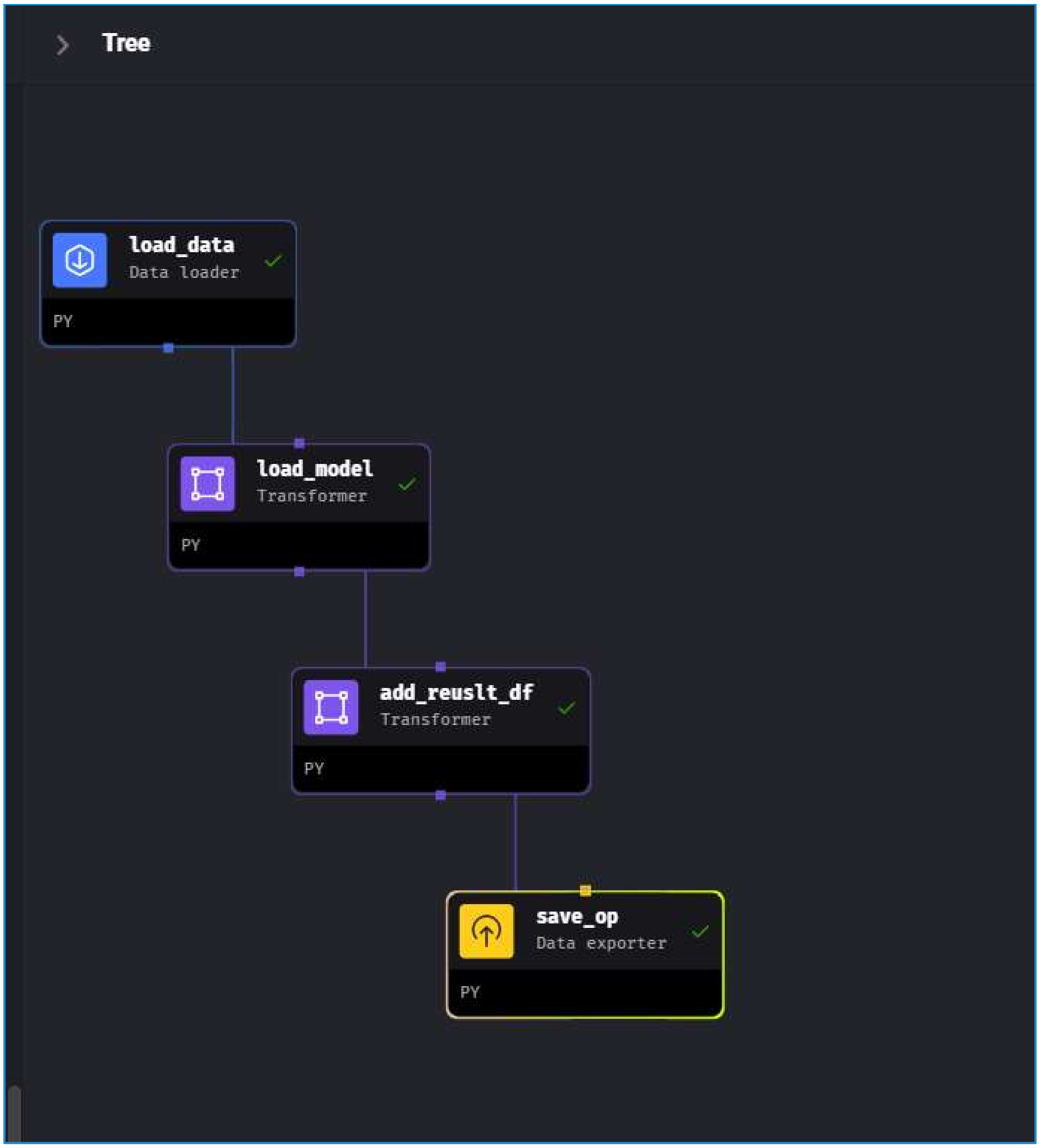
Find objects by prefix

□	Name	Type	Last modified	Size	Storage class
□	yellow_tripdata_2023-04.parquet	parquet	July 4, 2024, 20:00:34 (UTC+05:30)	63.1 MB	Standard
□	yellow_tripdata_2023-05.parquet	parquet	July 4, 2024, 20:16:24 (UTC+05:30)	67.1 MB	Standard

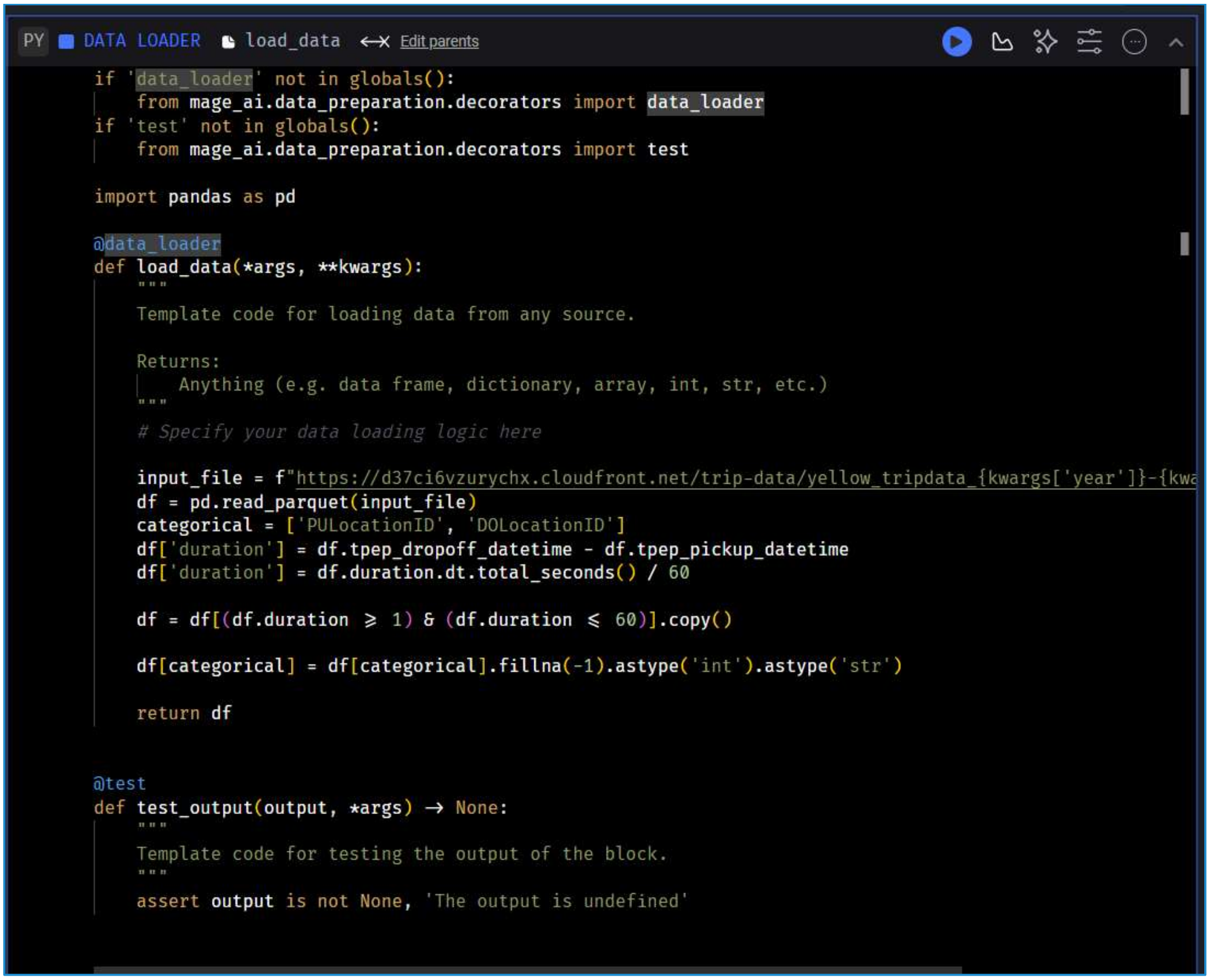
Bonus: Use Mage for batch inference

Here we didn't use any orchestration. In practice we usually do.

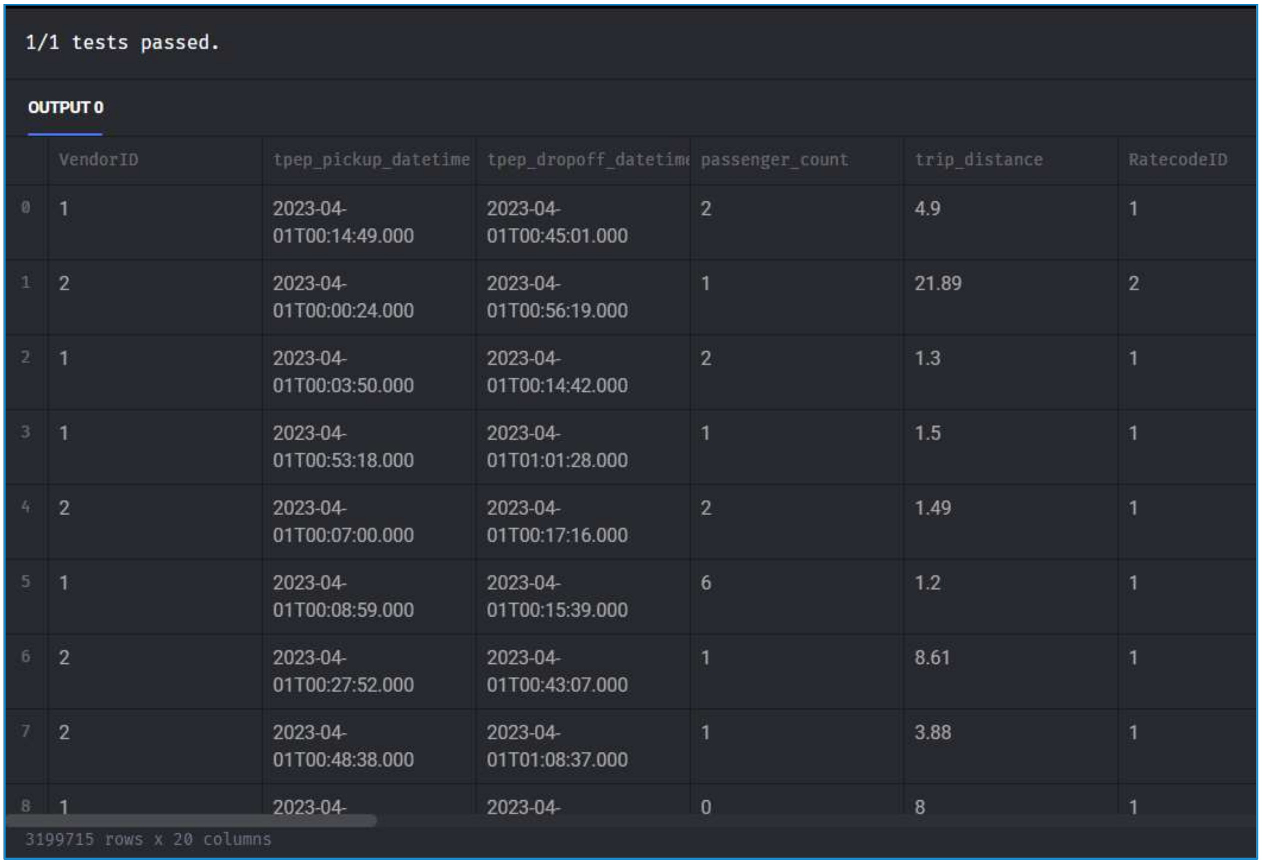
- Split the code into logical code blocks
- Use Mage to orchestrate the execution



- Creating data loader block to load data and add duration column



```
if 'data_loader' not in globals():
    from mage_ai.data_preparation.decorators import data_loader
if 'test' not in globals():
    from mage_ai.data_preparation.decorators import test

import pandas as pd

@data_loader
def load_data(*args, **kwargs):
    """
    Template code for loading data from any source.

    Returns:
        Anything (e.g. data frame, dictionary, array, int, str, etc.)
    """
    # Specify your data loading logic here

    input_file = f"https://d37ci6vzurychx.cloudfront.net/trip-data/yellow_tripdata_{kwargs['year']}-{kwa
    df = pd.read_parquet(input_file)
    categorical = ['PULocationID', 'DOLocationID']
    df['duration'] = df.tpep_dropoff_datetime - df.tpep_pickup_datetime
    df['duration'] = df.duration.dt.total_seconds() / 60

    df = df[(df.duration ⩾ 1) & (df.duration ⩽ 60)].copy()

    df[categorical] = df[categorical].fillna(-1).astype('int').astype('str')

    return df


@test
def test_output(output, *args) → None:
    """
    Template code for testing the output of the block.
    """
    assert output is not None, 'The output is undefined'
```



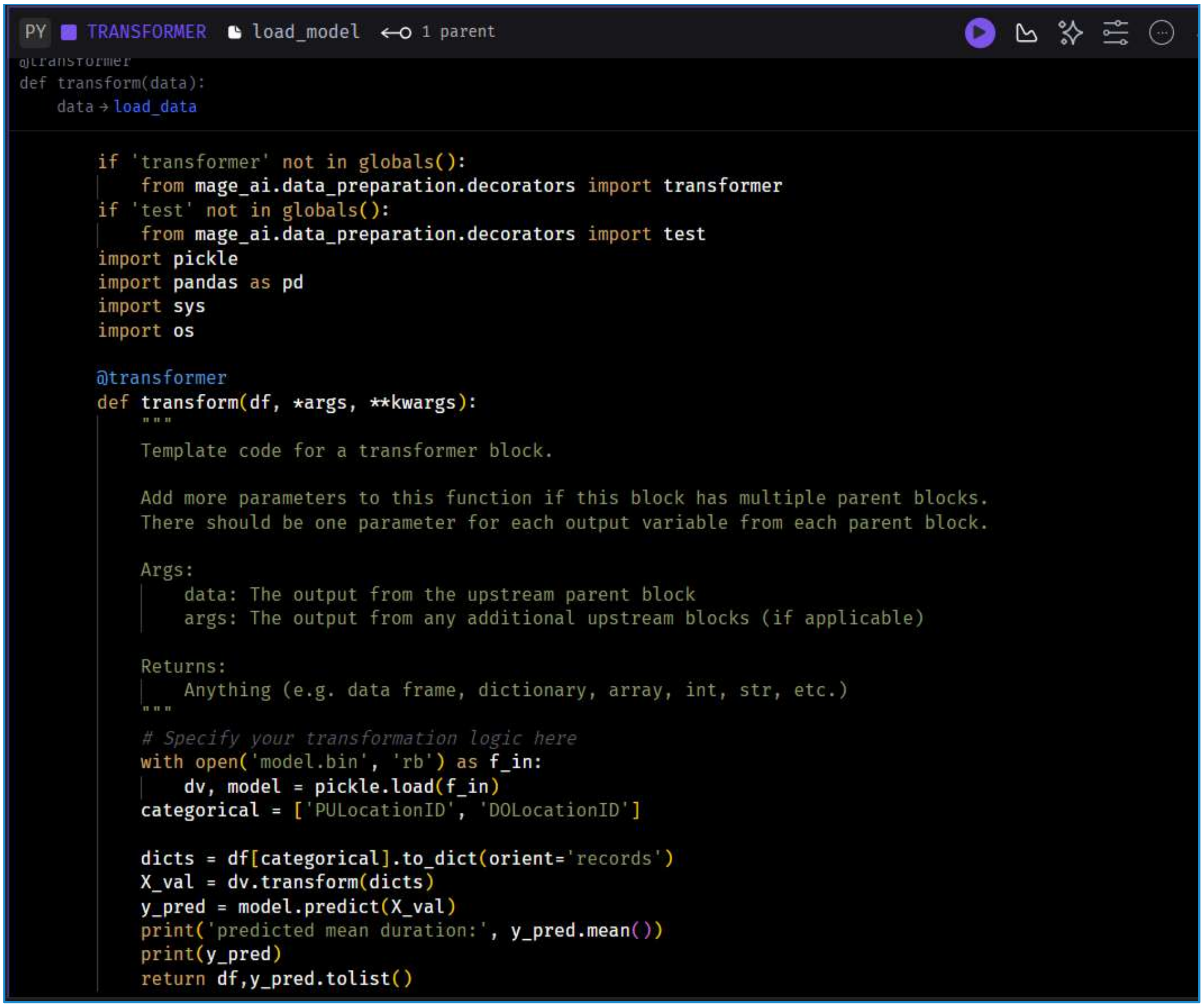
```
1/1 tests passed.
```

OUTPUT 0

	VendorID	tpep_pickup_datetime	tpep_dropoff_datetim	passenger_count	trip_distance	RatecodeID
0	1	2023-04-01T00:14:49.000	2023-04-01T00:45:01.000	2	4.9	1
1	2	2023-04-01T00:00:24.000	2023-04-01T00:56:19.000	1	21.89	2
2	1	2023-04-01T00:03:50.000	2023-04-01T00:14:42.000	2	1.3	1
3	1	2023-04-01T00:53:18.000	2023-04-01T01:01:28.000	1	1.5	1
4	2	2023-04-01T00:07:00.000	2023-04-01T00:17:16.000	2	1.49	1
5	1	2023-04-01T00:08:59.000	2023-04-01T00:15:39.000	6	1.2	1
6	2	2023-04-01T00:27:52.000	2023-04-01T00:43:07.000	1	8.61	1
7	2	2023-04-01T00:48:38.000	2023-04-01T01:08:37.000	1	3.88	1
8	1	2023-04-	2023-04-	0	8	1

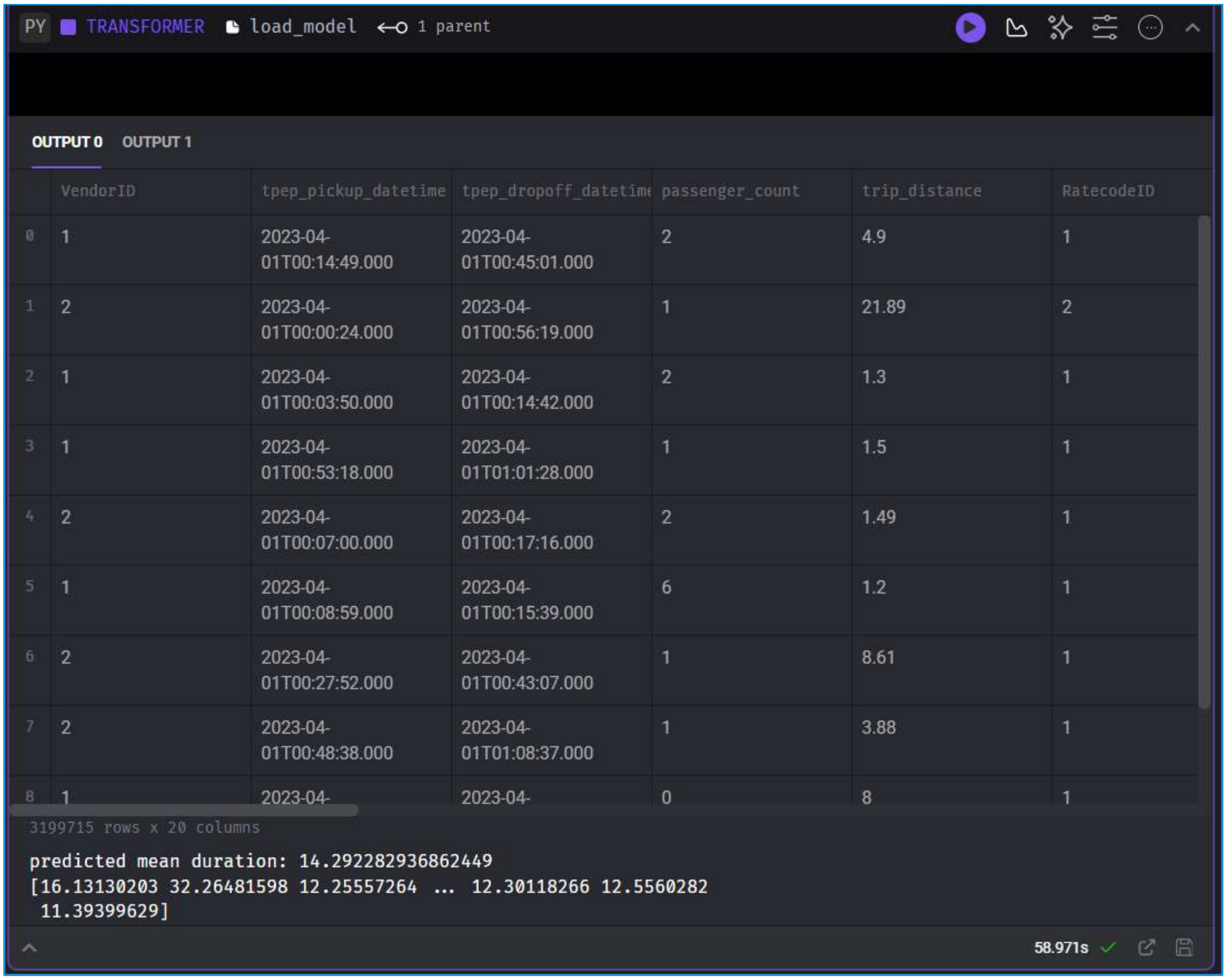
3199715 rows x 20 columns

- Adding transformation block to load the vectorizer and model, transforming the validation set and creating predictions.



```
@transformer
def transform(data):
    data → load_data
```

```
if 'transformer' not in globals():
    from mage_ai.data_preparation.decorators import transformer
if 'test' not in globals():
    from mage_ai.data_preparation.decorators import test
import pickle
import pandas as pd
import sys
import os

@transformer
def transform(df, *args, **kwargs):
    """
    Template code for a transformer block.

    Add more parameters to this function if this block has multiple parent blocks.
    There should be one parameter for each output variable from each parent block.

    Args:
        data: The output from the upstream parent block
        args: The output from any additional upstream blocks (if applicable)

    Returns:
        Anything (e.g. data frame, dictionary, array, int, str, etc.)
    """
    # Specify your transformation logic here
    with open('model.bin', 'rb') as f_in:
        dv, model = pickle.load(f_in)
    categorical = ['PULocationID', 'DOLocationID']

    dicts = df[categorical].to_dict(orient='records')
    X_val = dv.transform(dicts)
    y_pred = model.predict(X_val)
    print('predicted mean duration:', y_pred.mean())
    print(y_pred)
    return df,y_pred.tolist()
```

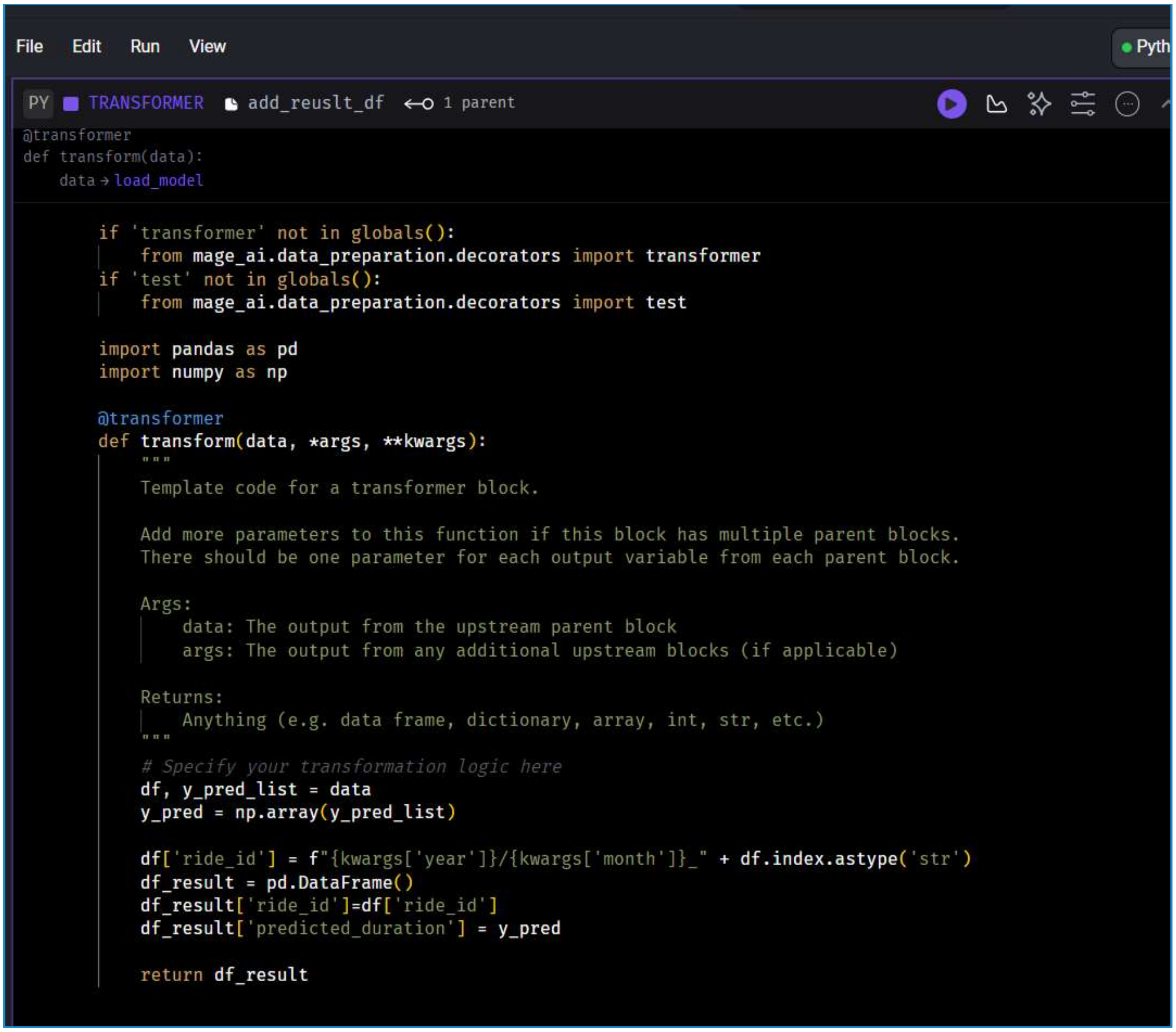


	VendorID	tpep_pickup_datetime	tpep_dropoff_datetim	passenger_count	trip_distance	RatecodeID
0	1	2023-04-01T00:14:49.000	2023-04-01T00:45:01.000	2	4.9	1
1	2	2023-04-01T00:00:24.000	2023-04-01T00:56:19.000	1	21.89	2
2	1	2023-04-01T00:03:50.000	2023-04-01T00:14:42.000	2	1.3	1
3	1	2023-04-01T00:53:18.000	2023-04-01T01:01:28.000	1	1.5	1
4	2	2023-04-01T00:07:00.000	2023-04-01T00:17:16.000	2	1.49	1
5	1	2023-04-01T00:08:59.000	2023-04-01T00:15:39.000	6	1.2	1
6	2	2023-04-01T00:27:52.000	2023-04-01T00:43:07.000	1	8.61	1
7	2	2023-04-01T00:48:38.000	2023-04-01T01:08:37.000	1	3.88	1
8	1	2023-04-	2023-04-	0	8	1

3199715 rows x 20 columns

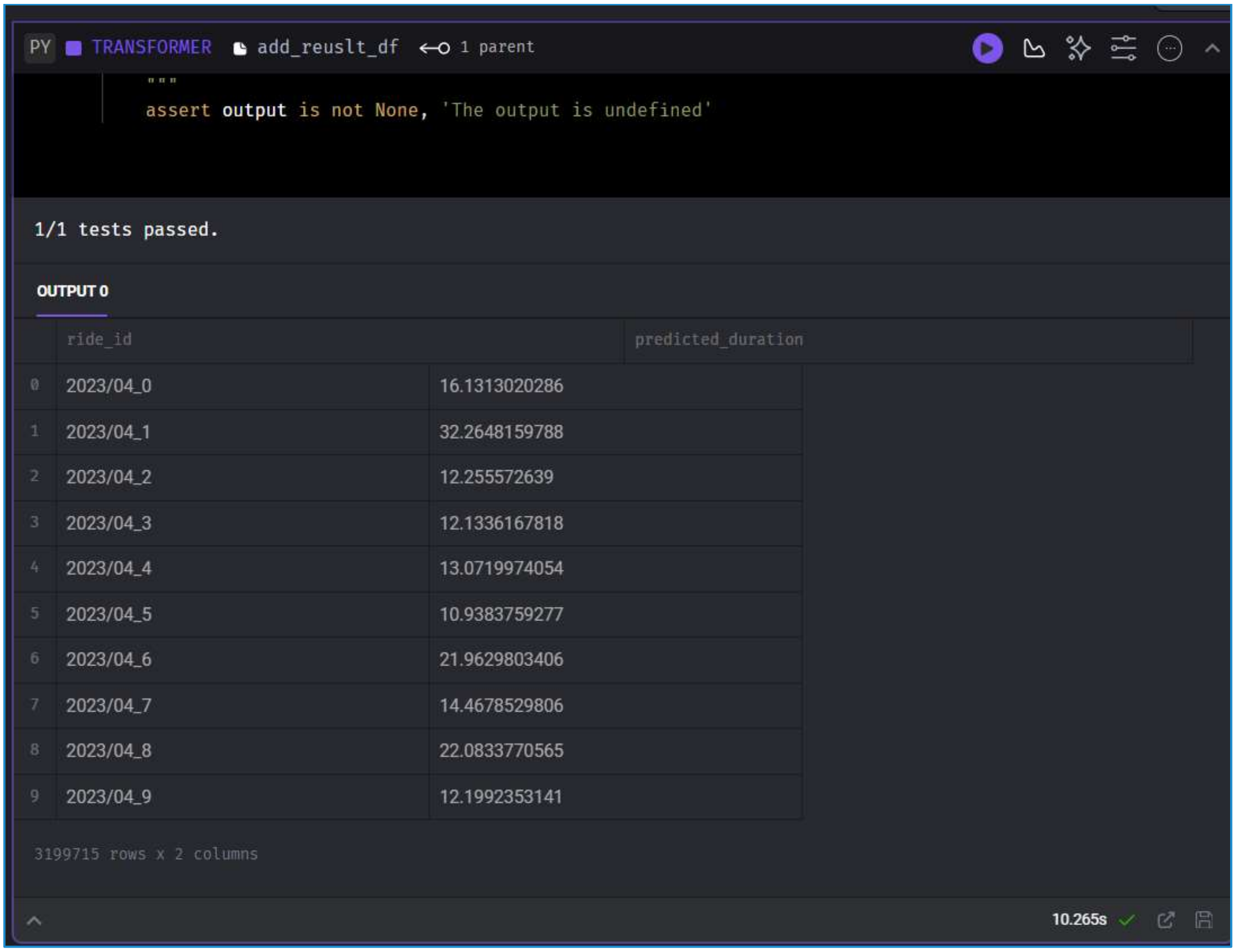
```
predicted mean duration: 14.292282936862449
[16.13130203 32.26481598 12.25557264 ... 12.30118266 12.5560282
 11.39399629]
```

- Adding df_result dataframe to store predictions



```
@transformer
def transform(data):
    data → load_model
```

```
if 'transformer' not in globals():
    from mage_ai.data_preparation.decorators import transformer
if 'test' not in globals():
    from mage_ai.data_preparation.decorators import test

import pandas as pd
import numpy as np

@transformer
def transform(data, *args, **kwargs):
    """
    Template code for a transformer block.

    Add more parameters to this function if this block has multiple parent blocks.
    There should be one parameter for each output variable from each parent block.

    Args:
        data: The output from the upstream parent block
        args: The output from any additional upstream blocks (if applicable)

    Returns:
        Anything (e.g. data frame, dictionary, array, int, str, etc.)
    """
    # Specify your transformation logic here
    df, y_pred_list = data
    y_pred = np.array(y_pred_list)

    df['ride_id'] = f"{kwargs['year']}/{kwargs['month']}_" + df.index.astype('str')
    df_result = pd.DataFrame()
    df_result['ride_id']=df['ride_id']
    df_result['predicted_duration'] = y_pred

    return df_result
```



```
    """
    assert output is not None, 'The output is undefined'
```

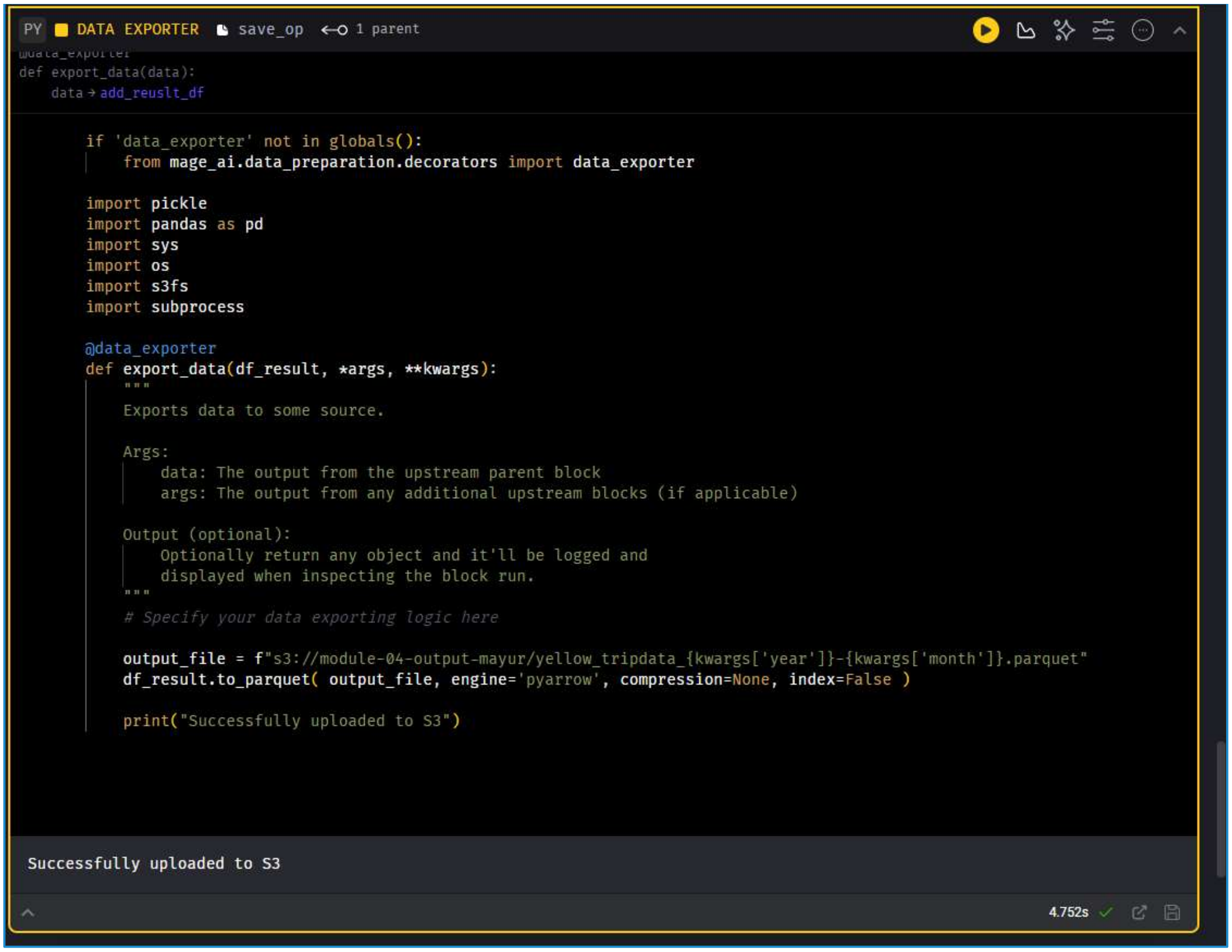
```
1/1 tests passed.
```

	ride_id	predicted_duration
0	2023/04_0	16.1313020286
1	2023/04_1	32.2648159788
2	2023/04_2	12.255572639
3	2023/04_3	12.1336167818
4	2023/04_4	13.0719974054
5	2023/04_5	10.9383759277
6	2023/04_6	21.9629803406
7	2023/04_7	14.4678529806
8	2023/04_8	22.0833770565
9	2023/04_9	12.1992353141

3199715 rows x 2 columns

10.265s ✓

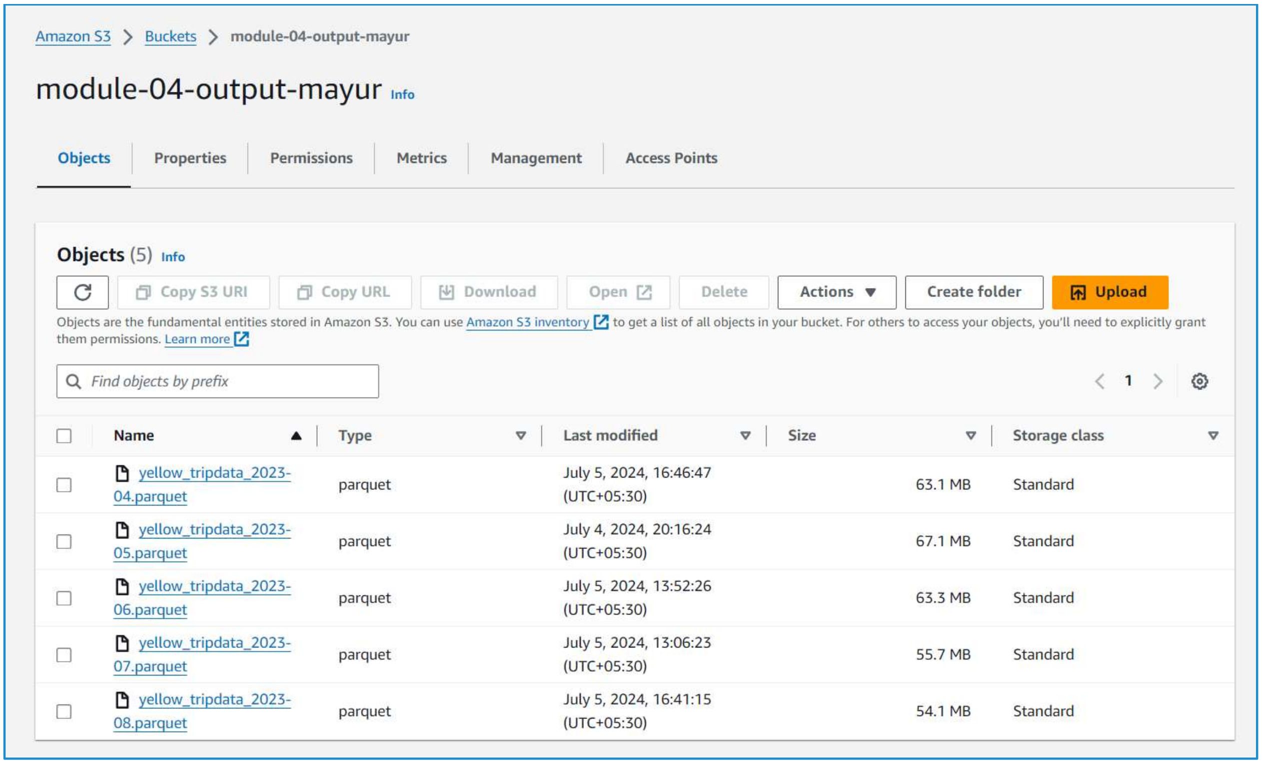
- Adding data exporter block to store results into S3 Bucket



```
@data_exporter
def export_data(data):
    data → add_reuslt_df
```

```python
if 'data_exporter' not in globals():
    from mage_ai.data_preparation.decorators import data_exporter

import pickle
import pandas as pd
import sys
import os
import s3fs
import subprocess

@data_exporter
def export_data(df_result, *args, **kwargs):
    """
    Exports data to some source.

    Args:
        data: The output from the upstream parent block
        args: The output from any additional upstream blocks (if applicable)

    Output (optional):
        Optionally return any object and it'll be logged and
        displayed when inspecting the block run.
    """
    # Specify your data exporting logic here

    output_file = f"s3://module-04-output-mayur/yellow_tripdata_{kwargs['year']}-{kwargs['month']}.parquet"
    df_result.to_parquet( output_file, engine='pyarrow', compression=None, index=False )

    print("Successfully uploaded to S3")
```

```
Successfully uploaded to S3
```



module-04-output-mayur

Objects (5)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

□	Name	Type	Last modified	Size	Storage class
□	yellow_tripdata_2023-04.parquet	parquet	July 5, 2024, 16:46:47 (UTC+05:30)	63.1 MB	Standard
□	yellow_tripdata_2023-05.parquet	parquet	July 4, 2024, 20:16:24 (UTC+05:30)	67.1 MB	Standard
□	yellow_tripdata_2023-06.parquet	parquet	July 5, 2024, 13:52:26 (UTC+05:30)	63.3 MB	Standard
□	yellow_tripdata_2023-07.parquet	parquet	July 5, 2024, 13:06:23 (UTC+05:30)	55.7 MB	Standard
□	yellow_tripdata_2023-08.parquet	parquet	July 5, 2024, 16:41:15 (UTC+05:30)	54.1 MB	Standard