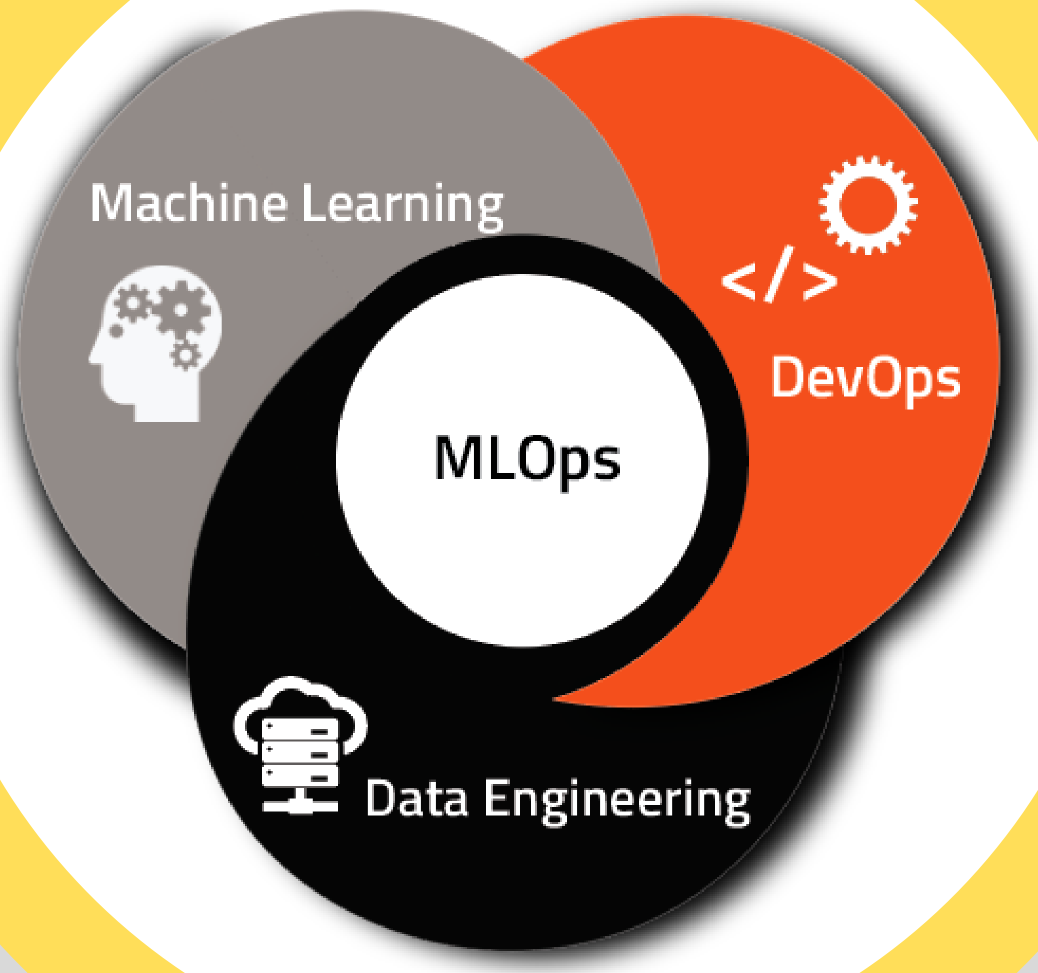
StatusNeo

statusneo.com



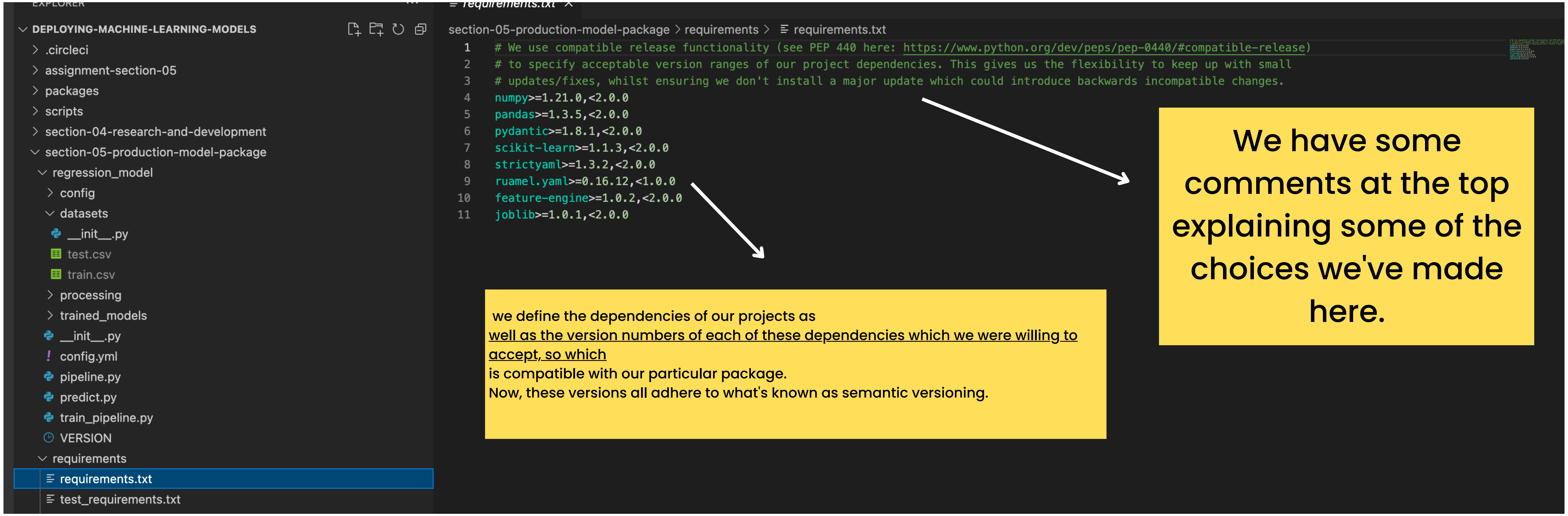
MLOps

Packaging Model for Production

SESSION 5

StatusNeo

Requirements File



EXPLORER
requirements.txt
section-05-production-model-package > requirements > requirements.txt

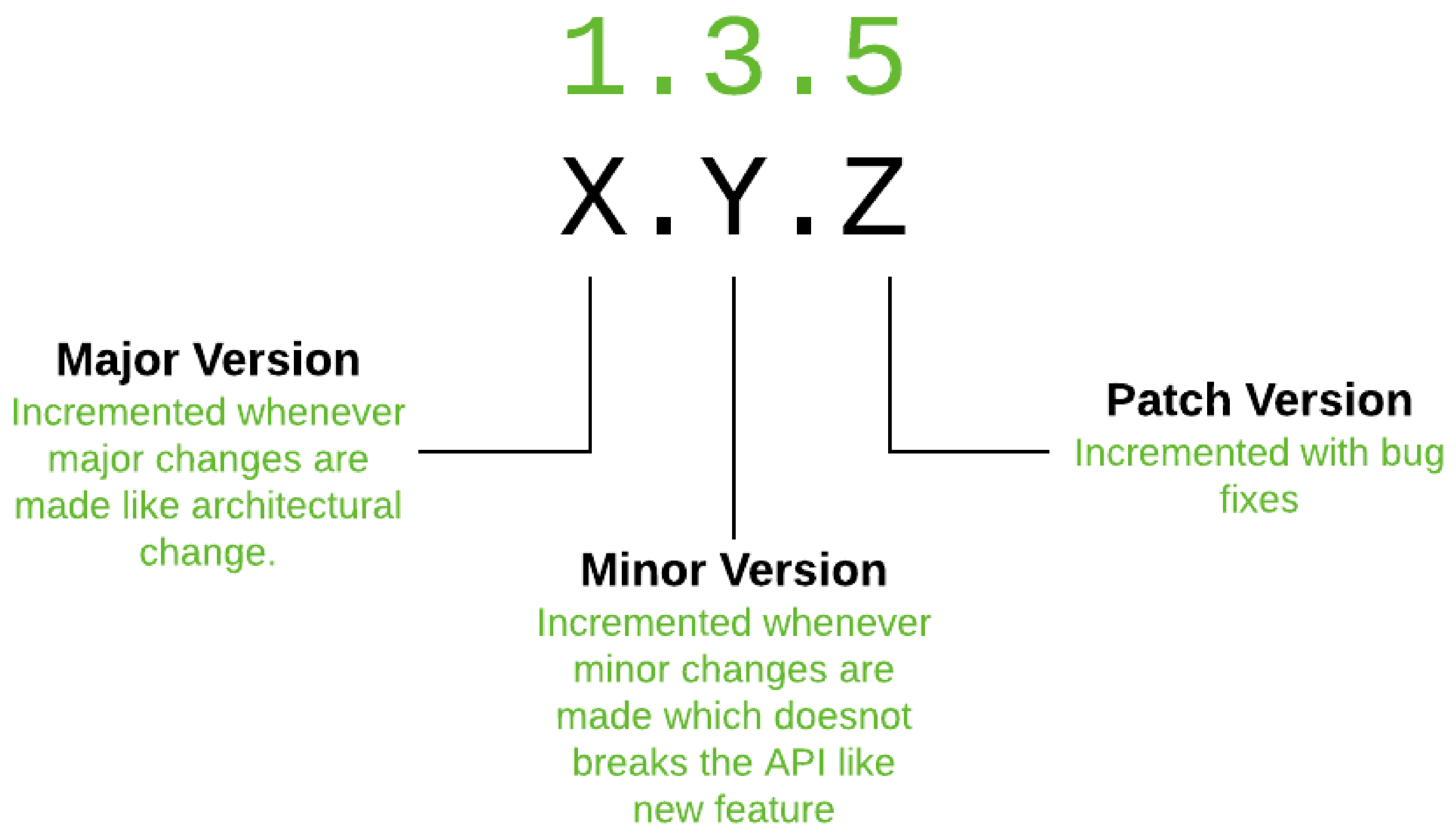
```
DEPLOYING-MACHINE-LEARNING-MODELS
  > .circleci
  > assignment-section-05
  > packages
  > scripts
  > section-04-research-and-development
  v section-05-production-model-package
    v regression_model
      > config
      v datasets
          __init__.py
          test.csv
          train.csv
      > processing
      > trained_models
        __init__.py
        config.yml
        pipeline.py
        predict.py
        train_pipeline.py
        VERSION
    v requirements
        requirements.txt
        test_requirements.txt
```

```
# We use compatible release functionality (see PEP 440 here: https://www.python.org/dev/peps/pep-0440/#compatible-release)
# to specify acceptable version ranges of our project dependencies. This gives us the flexibility to keep up with small
# updates/fixes, whilst ensuring we don't install a major update which could introduce backwards incompatible changes.
numpy>=1.21.0,<2.0.0
pandas>=1.3.5,<2.0.0
pydantic>=1.8.1,<2.0.0
scikit-learn>=1.1.3,<2.0.0
strictyaml>=1.3.2,<2.0.0
ruamel.yaml>=0.16.12,<1.0.0
feature-engine>=1.0.2,<2.0.0
joblib>=1.0.1,<2.0.0
```

We have some comments at the top explaining some of the choices we've made here.

we define the dependencies of our projects as
well as the version numbers of each of these dependencies which we were willing to accept, so which
is compatible with our particular package.
Now, these versions all adhere to what's known as semantic versioning.

Semantic Versions


1.3.5
X.Y.Z
Major Version
Incremented whenever major changes are made like architectural change.
Minor Version
Incremented whenever minor changes are made which doesnot breaks the API like new feature
Patch Version
Incremented with bug fixes

You have a major version followed by a period followed by the minor version, followed by a period followed

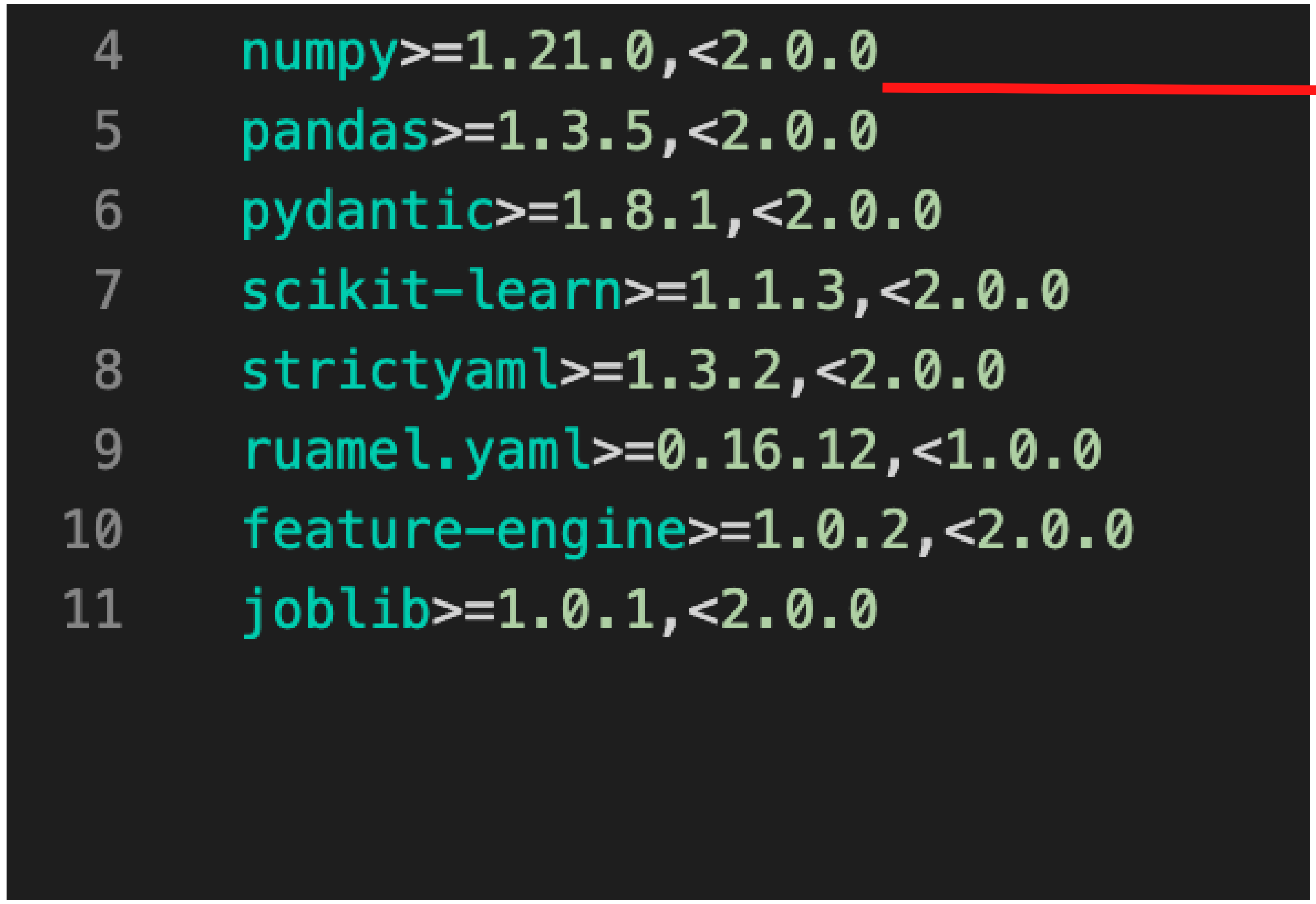
by what's known as a patch version.

And for a well maintained package, you'd expect that a minor version increment does not break the API and a major version increment is likely to break the API.

But you have to be a bit careful because some less well maintain packages may actually introduce breaking changes, even in a minor version bump.

StatusNeo

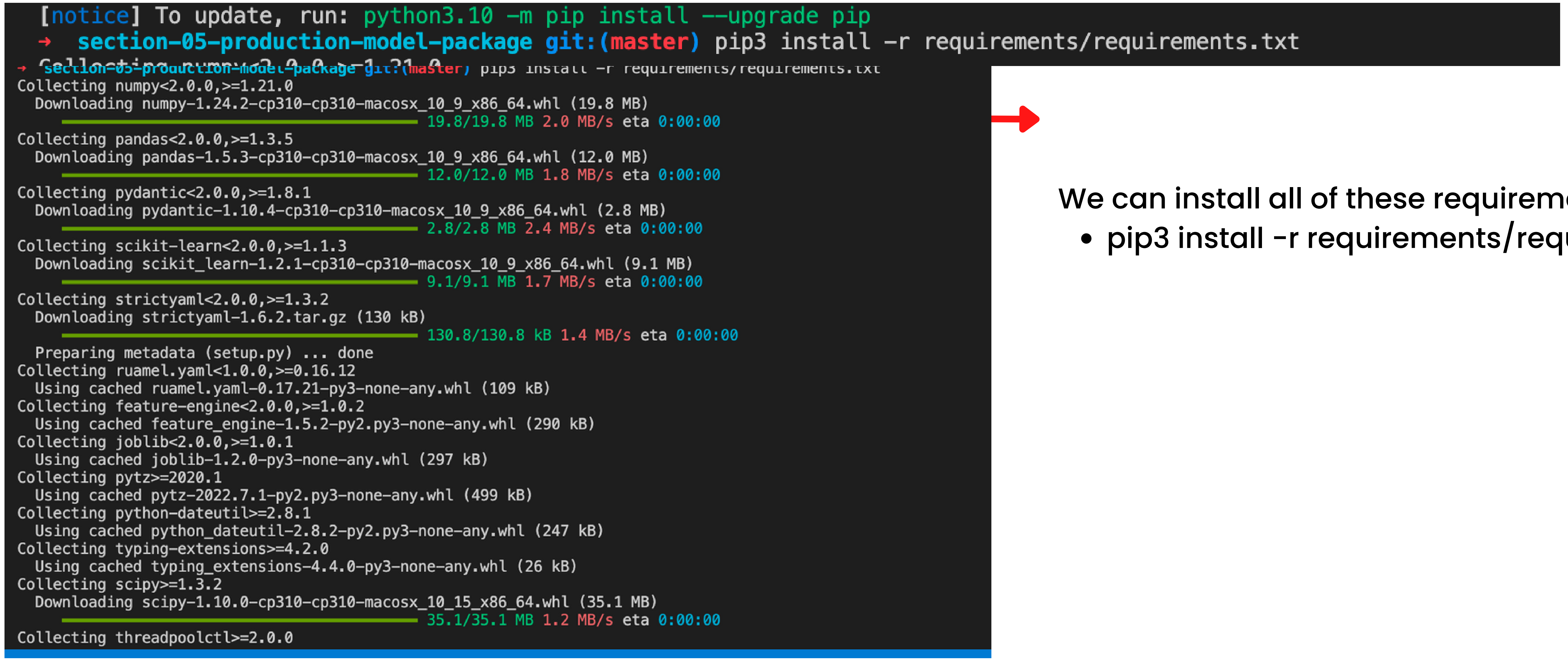
Play it Conservatively!


```
numpy>=1.21.0,<2.0.0
pandas>=1.3.5,<2.0.0
pydantic>=1.8.1,<2.0.0
scikit-learn>=1.1.3,<2.0.0
strictyaml>=1.3.2,<2.0.0
ruamel.yaml>=0.16.12,<1.0.0
feature-engine>=1.0.2,<2.0.0
joblib>=1.0.1,<2.0.0
```

We're saying that anything for no greater than version one point to zero point zero, but it has to be less than version one point two one.
So we're not allowing any minor version increases here.
And it's up to you and your projects, how much risk you want to take.

StatusNeo

Play it Conservatively!



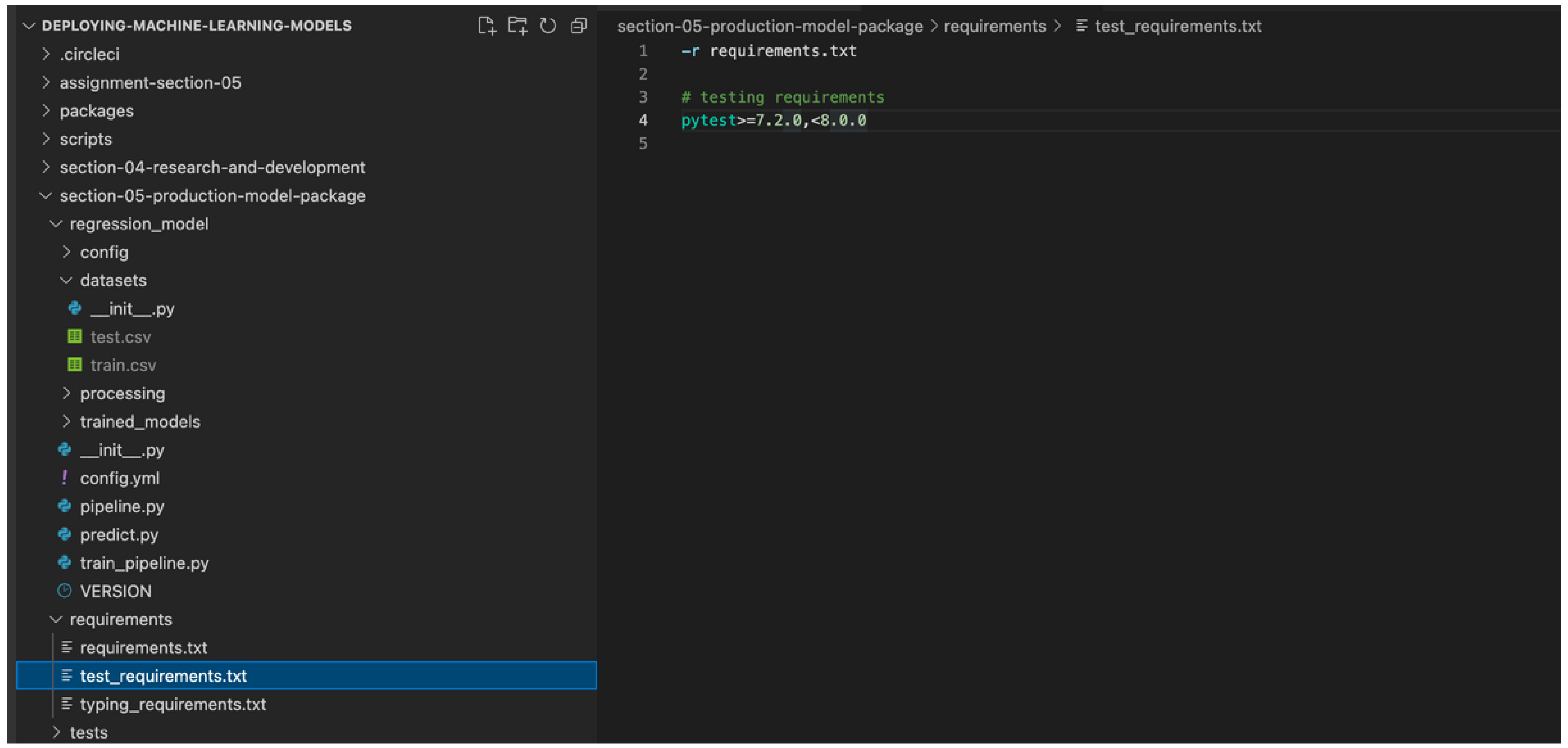
```
[notice] To update, run: python3.10 -m pip install --upgrade pip
➜  section-05-production-model-package git:(master) pip3 install -r requirements/requirements.txt
➜  section-05-production-model-package git:(master) pip3 install -r requirements/requirements.txt
Collecting numpy<2.0.0,>=1.21.0
  Downloading numpy-1.24.2-cp310-cp310-macosx_10_9_x86_64.whl (19.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 19.8/19.8 MB 2.0 MB/s eta 0:00:00
Collecting pandas<2.0.0,>=1.3.5
  Downloading pandas-1.5.3-cp310-cp310-macosx_10_9_x86_64.whl (12.0 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.0/12.0 MB 1.8 MB/s eta 0:00:00
Collecting pydantic<2.0.0,>=1.8.1
  Downloading pydantic-1.10.4-cp310-cp310-macosx_10_9_x86_64.whl (2.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.8/2.8 MB 2.4 MB/s eta 0:00:00
Collecting scikit-learn<2.0.0,>=1.1.3
  Downloading scikit_learn-1.2.1-cp310-cp310-macosx_10_9_x86_64.whl (9.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.1/9.1 MB 1.7 MB/s eta 0:00:00
Collecting strictyaml<2.0.0,>=1.3.2
  Downloading strictyaml-1.6.2.tar.gz (130 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 130.8/130.8 kB 1.4 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Collecting ruamel.yaml<1.0.0,>=0.16.12
  Using cached ruamel.yaml-0.17.21-py3-none-any.whl (109 kB)
Collecting feature-engine<2.0.0,>=1.0.2
  Using cached feature_engine-1.5.2-py2.py3-none-any.whl (290 kB)
Collecting joblib<2.0.0,>=1.0.1
  Using cached joblib-1.2.0-py3-none-any.whl (297 kB)
Collecting pytz>=2020.1
  Using cached pytz-2022.7.1-py2.py3-none-any.whl (499 kB)
Collecting python-dateutil>=2.8.1
  Using cached python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
Collecting typing-extensions>=4.2.0
  Using cached typing_extensions-4.4.0-py3-none-any.whl (26 kB)
Collecting scipy>=1.3.2
  Downloading scipy-1.10.0-cp310-cp310-macosx_10_15_x86_64.whl (35.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 35.1/35.1 MB 1.2 MB/s eta 0:00:00
Collecting threadpoolctl>=2.0.0
```

We can install all of these requirements with the commands

- pip3 install -r requirements/requirements.txt

StatusNeo

Another requirement file : test_requirements.txt



```
-r requirements.txt

# testing requirements
pytest>=7.2.0,<8.0.0

```

So it's a way of just capturing everything in another requirements file. And we've split the requirements into these two files because there will be scenarios where we don't actually need to install these test requirements because these are only required when we want to test our package or when we want to run style checks, linting and type checks.

StatusNeo

Why it is important to define a version

- These two files are really important. If we don't define which versions of our dependencies we expect, then it can result in our package becoming very brittle and broken.
- The most basic error in a package would be that you didn't have a requirement to file.
- That would just mean that when somebody else tried to install and use the package, it would fail because it wouldn't have its necessary dependencies.
- A more likely scenario is that we neglect to include a version, and if there is no version specified, PIP is just going to assume you want the latest version of a particular dependency.
- And it may well be that that version has progressed and it's released new features, a new major version perhaps, and it's got breaking changes in the API.
- And that means that your package or our package in this case is going to break.So it's really important for us to define these versions.

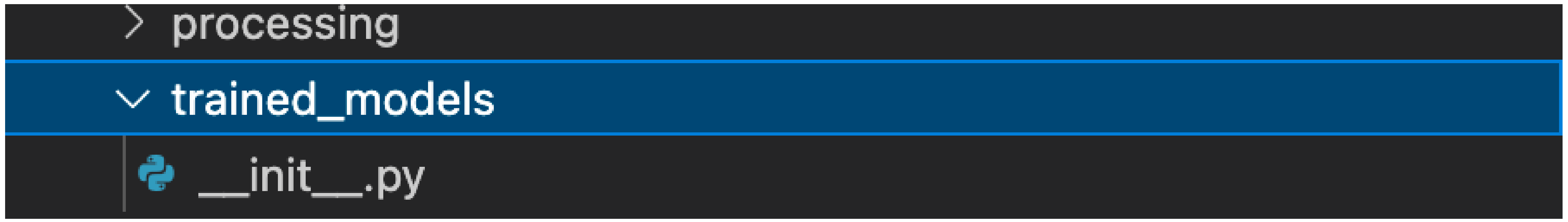
What is Tox? How does it work?

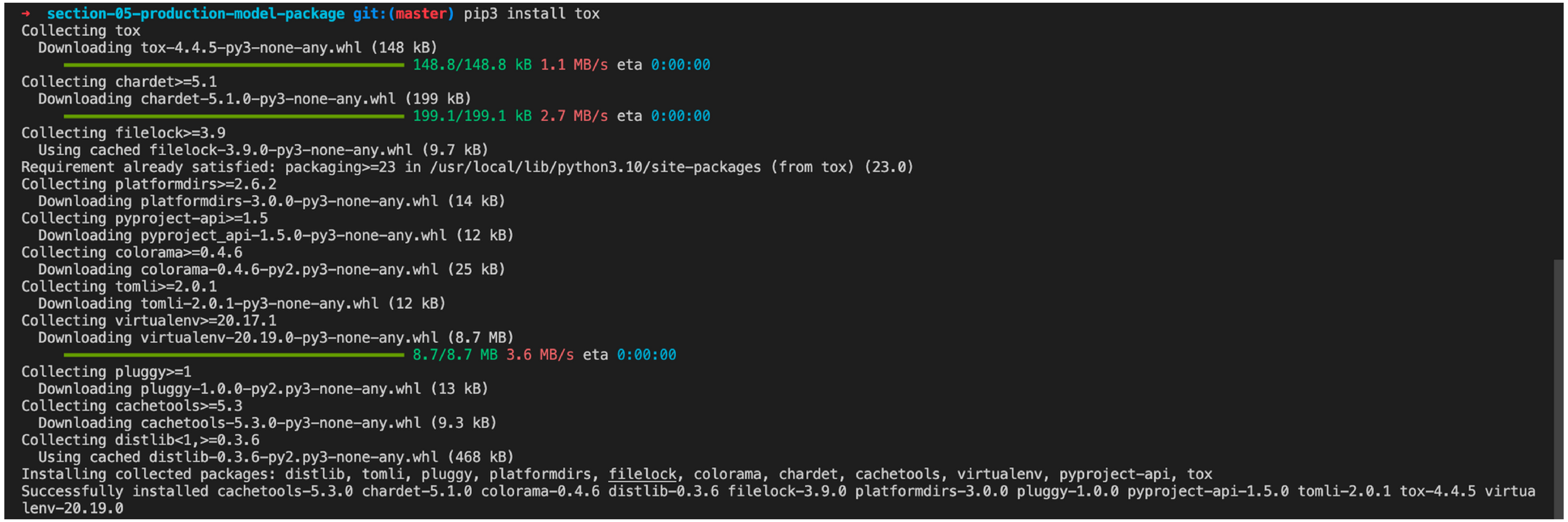
As you can see, Tox is generic virtual environment management and test command line tool.

- So what that really means for our purposes here is that tox means we don't have to worry about different operating systems. We can run tox on Windows, macOS, Linux and get the same behavior across the platforms.
- We don't have to worry about things like setting up python paths, configuring environment variables.
- We do all of that stuff inside our tox ini file. So it's a really powerful way for us to be able to run tests of different versions of Python.

https://tox.wiki/en/latest/

If you go train model , currently you just see _ _init_ _.py file. Now headover to command prompt and install tox

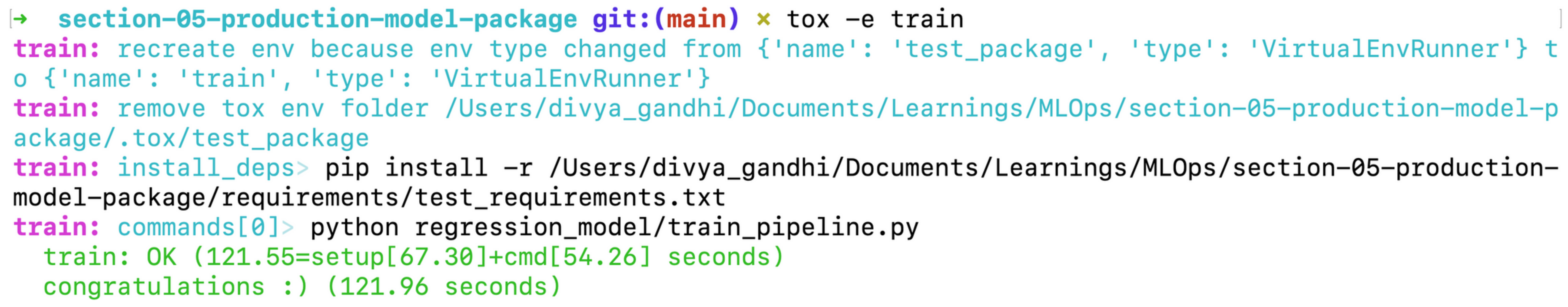




```
➜  section-05-production-model-package git:(master) pip3 install tox
Collecting tox
  Downloading tox-4.4.5-py3-none-any.whl (148 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 148.8/148.8 kB 1.1 MB/s eta 0:00:00
Collecting chardet>=5.1
  Downloading chardet-5.1.0-py3-none-any.whl (199 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 199.1/199.1 kB 2.7 MB/s eta 0:00:00
Collecting filelock>=3.9
  Using cached filelock-3.9.0-py3-none-any.whl (9.7 kB)
Requirement already satisfied: packaging>=23 in /usr/local/lib/python3.10/site-packages (from tox) (23.0)
Collecting platformdirs>=2.6.2
  Downloading platformdirs-3.0.0-py3-none-any.whl (14 kB)
Collecting pyproject-api>=1.5
  Downloading pyproject_api-1.5.0-py3-none-any.whl (12 kB)
Collecting colorama>=0.4.6
  Downloading colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Collecting tomli>=2.0.1
  Downloading tomli-2.0.1-py3-none-any.whl (12 kB)
Collecting virtualenv>=20.17.1
  Downloading virtualenv-20.19.0-py3-none-any.whl (8.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.7/8.7 MB 3.6 MB/s eta 0:00:00
Collecting pluggy>=1
  Downloading pluggy-1.0.0-py2.py3-none-any.whl (13 kB)
Collecting cachetools>=5.3
  Downloading cachetools-5.3.0-py3-none-any.whl (9.3 kB)
Collecting distlib<1,>=0.3.6
  Using cached distlib-0.3.6-py2.py3-none-any.whl (468 kB)
Installing collected packages: distlib, tomli, pluggy, platformdirs, filelock, colorama, chardet, cachetools, virtualenv, pyproject-api, tox
Successfully installed cachetools-5.3.0 chardet-5.1.0 colorama-0.4.6 distlib-0.3.6 filelock-3.9.0 platformdirs-3.0.0 pluggy-1.0.0 pyproject-api-1.5.0 tomli-2.0.1 tox-4.4.5 virtua
lenv-20.19.0
```

StatusNeo

Run the tox command

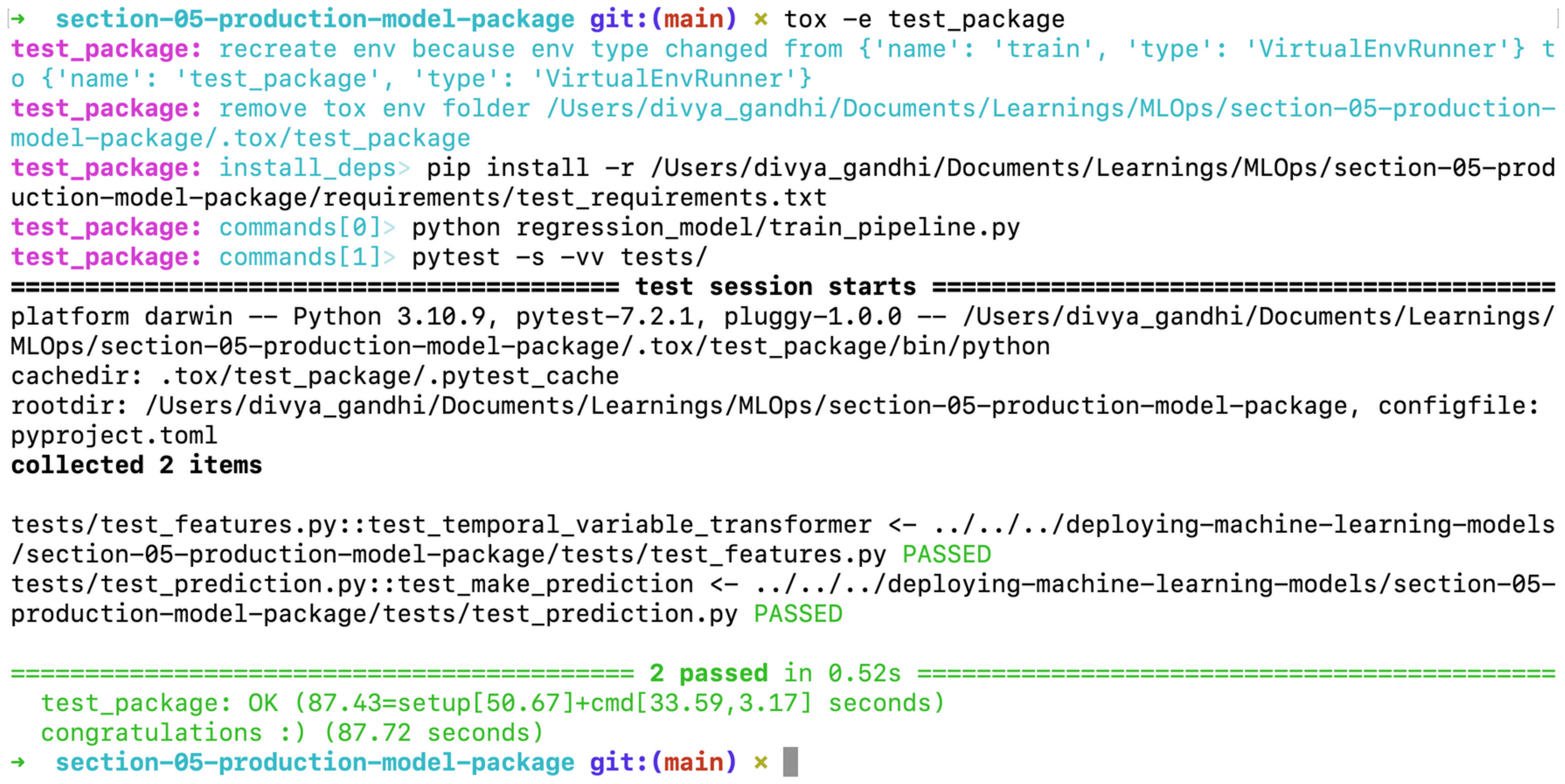


```
➜  section-05-production-model-package git:(main) ✗ tox -e train
train: recreate env because env type changed from {'name': 'test_package', 'type': 'VirtualEnvRunner'} t
o {'name': 'train', 'type': 'VirtualEnvRunner'}
train: remove tox env folder /Users/divya_gandhi/Documents/Learnings/MLOps/section-05-production-model-p
ackage/.tox/test_package
train: install_deps> pip install -r /Users/divya_gandhi/Documents/Learnings/MLOps/section-05-production-
model-package/requirements/test_requirements.txt
train: commands[0]> python regression_model/train_pipeline.py
  train: OK (121.55=setup[67.30]+cmd[54.26] seconds)
  congratulations :) (121.96 seconds)
```

Run the command tox -e train , and as you see it is succeeded. -e means I want to specify the environment

StatusNeo

Run the tox command for test_package



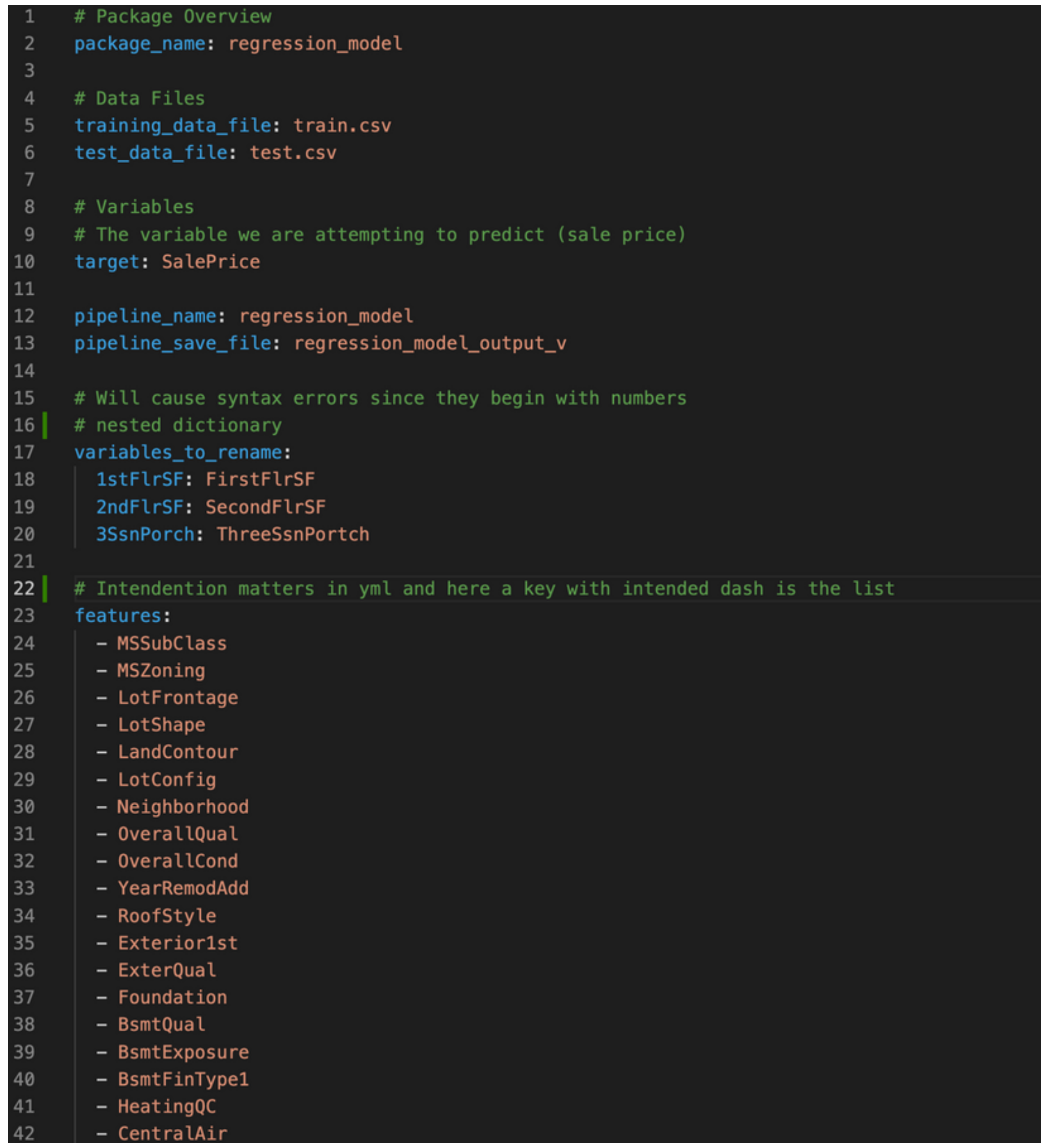
```
→  section-05-production-model-package git:(main) ✗ tox -e test_package
test_package: recreate env because env type changed from {'name': 'train', 'type': 'VirtualEnvRunner'} t
o {'name': 'test_package', 'type': 'VirtualEnvRunner'}
test_package: remove tox env folder /Users/divya_gandhi/Documents/Learnings/MLOps/section-05-production-
model-package/.tox/test_package
test_package: install_deps> pip install -r /Users/divya_gandhi/Documents/Learnings/MLOps/section-05-prod
uction-model-package/requirements/test_requirements.txt
test_package: commands[0]> python regression_model/train_pipeline.py
test_package: commands[1]> pytest -s -vv tests/
========================================= test session starts ==========================================
platform darwin -- Python 3.10.9, pytest-7.2.1, pluggy-1.0.0 -- /Users/divya_gandhi/Documents/Learnings/
MLOps/section-05-production-model-package/.tox/test_package/bin/python
cachedir: .tox/test_package/.pytest_cache
rootdir: /Users/divya_gandhi/Documents/Learnings/MLOps/section-05-production-model-package, configfile:
pyproject.toml
collected 2 items

tests/test_features.py::test_temporal_variable_transformer <- ../../../deploying-machine-learning-models
/section-05-production-model-package/tests/test_features.py PASSED
tests/test_prediction.py::test_make_prediction <- ../../../deploying-machine-learning-models/section-05-
production-model-package/tests/test_prediction.py PASSED

========================================== 2 passed in 0.52s ===========================================
  test_package: OK (87.43=setup[50.67]+cmd[33.59,3.17] seconds)
  congratulations :) (87.72 seconds)
→  section-05-production-model-package git:(main) ✗
```

Out here if you see two tests are run as mentioned in our test folder:

StatusNeo

Config File

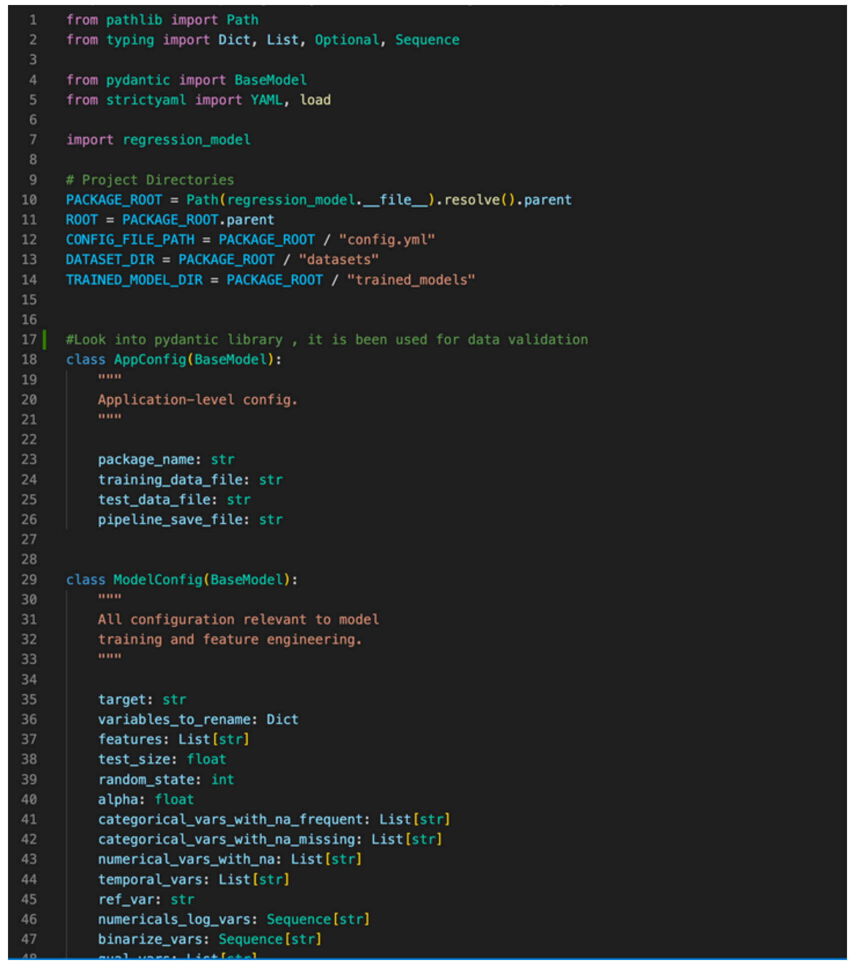
Why use config.yml rather than python file. Check out the link mentioned on REAME.md file



```yaml
# Package Overview
package_name: regression_model

# Data Files
training_data_file: train.csv
test_data_file: test.csv

# Variables
# The variable we are attempting to predict (sale price)
target: SalePrice

pipeline_name: regression_model
pipeline_save_file: regression_model_output_v

# Will cause syntax errors since they begin with numbers
# nested dictionary
variables_to_rename:
  1stFlrSF: FirstFlrSF
  2ndFlrSF: SecondFlrSF
  3SsnPorch: ThreeSsnPortch

# Intendention matters in yml and here a key with intended dash is the list
features:
  - MSSubClass
  - MSZoning
  - LotFrontage
  - LotShape
  - LandContour
  - LotConfig
  - Neighborhood
  - OverallQual
  - OverallCond
  - YearRemodAdd
  - RoofStyle
  - Exterior1st
  - ExterQual
  - Foundation
  - BsmtQual
  - BsmtExposure
  - BsmtFinType1
  - HeatingQC
  - CentralAir
```

pathlib for defining the location of config yml are defining path of pacakge root, config file path, dataset path

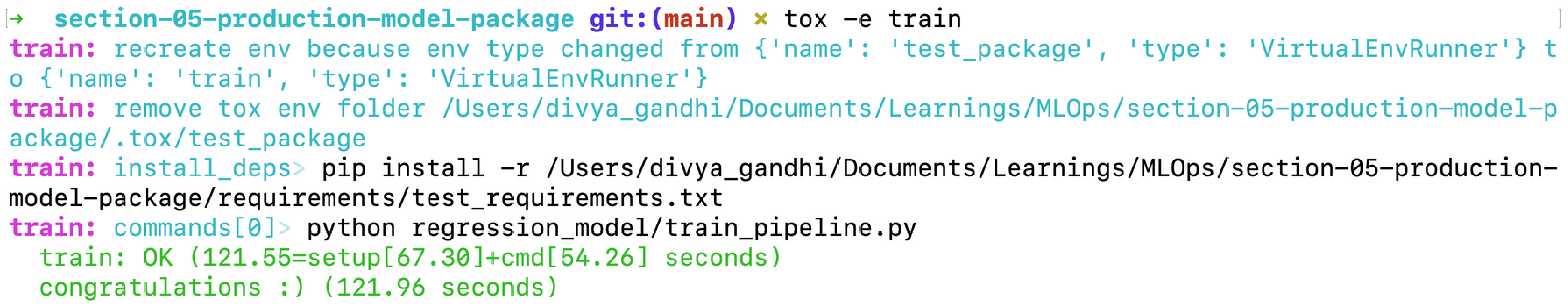
pydantic library is used to define the types of each column.



```python
from pathlib import Path
from typing import Dict, List, Optional, Sequence

from pydantic import BaseModel
from strictyaml import YAML, load

import regression_model

# Project Directories
PACKAGE_ROOT = Path(regression_model.__file__).resolve().parent
ROOT = PACKAGE_ROOT.parent
CONFIG_FILE_PATH = PACKAGE_ROOT / "config.yml"
DATASET_DIR = PACKAGE_ROOT / "datasets"
TRAINED_MODEL_DIR = PACKAGE_ROOT / "trained_models"


#Look into pydantic library , it is been used for data validation
class AppConfig(BaseModel):
    """
    Application-level config.
    """

    package_name: str
    training_data_file: str
    test_data_file: str
    pipeline_save_file: str


class ModelConfig(BaseModel):
    """
    All configuration relevant to model
    training and feature engineering.
    """

    target: str
    variables_to_rename: Dict
    features: List[str]
    test_size: float
    random_state: int
    alpha: float
    categorical_vars_with_na_frequent: List[str]
    categorical_vars_with_na_missing: List[str]
    numerical_vars_with_na: List[str]
    temporal_vars: List[str]
    ref_var: str
    numericals_log_vars: Sequence[str]
    binarize_vars: Sequence[str]
```

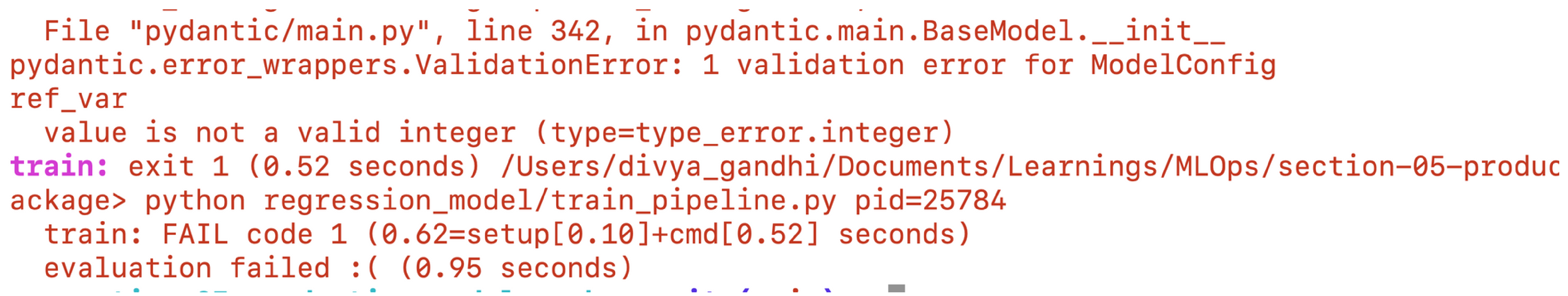
StatusNeo

Lets Test Incorrect Config



```
→  section-05-production-model-package git:(main) × tox -e train
train: recreate env because env type changed from {'name': 'test_package', 'type': 'VirtualEnvRunner'} t
o {'name': 'train', 'type': 'VirtualEnvRunner'}
train: remove tox env folder /Users/divya_gandhi/Documents/Learnings/MLOps/section-05-production-model-p
ackage/.tox/test_package
train: install_deps> pip install -r /Users/divya_gandhi/Documents/Learnings/MLOps/section-05-production-
model-package/requirements/test_requirements.txt
train: commands[0]> python regression_model/train_pipeline.py
  train: OK (121.55=setup[67.30]+cmd[54.26] seconds)
  congratulations :) (121.96 seconds)
```

tox -e train successful run

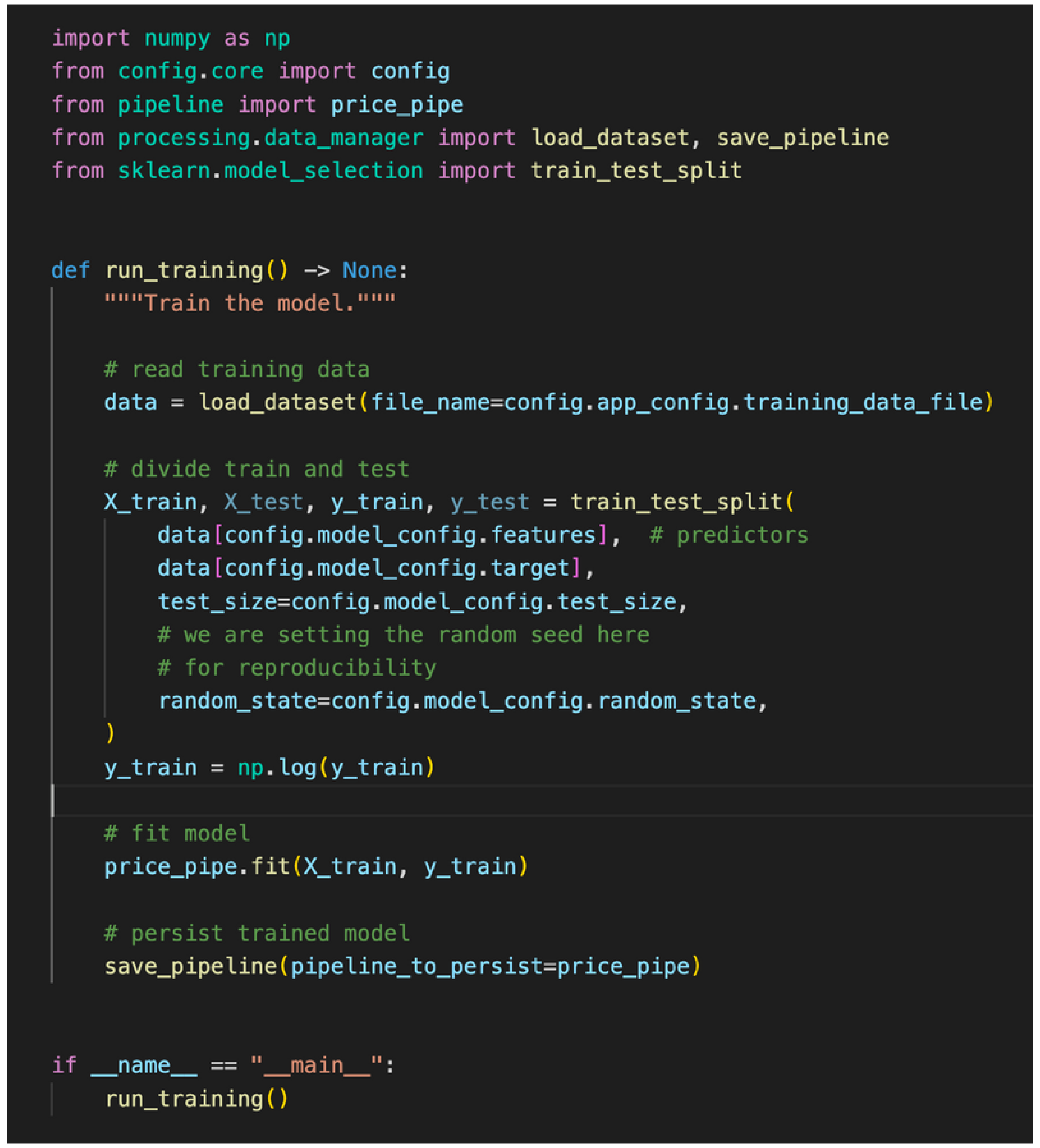


```
  File "pydantic/main.py", line 342, in pydantic.main.BaseModel.__init__
pydantic.error_wrappers.ValidationError: 1 validation error for ModelConfig
ref_var
  value is not a valid integer (type=type_error.integer)
train: exit 1 (0.52 seconds) /Users/divya_gandhi/Documents/Learnings/MLOps/section-05-produc
ackage> python regression_model/train_pipeline.py pid=25784
  train: FAIL code 1 (0.62=setup[0.10]+cmd[0.52] seconds)
  evaluation failed :( (0.95 seconds)
```

tox -e train unsuccessful run after changing ref_var type to int

StatusNeo

Understand Train Pipeline and Pipeline module

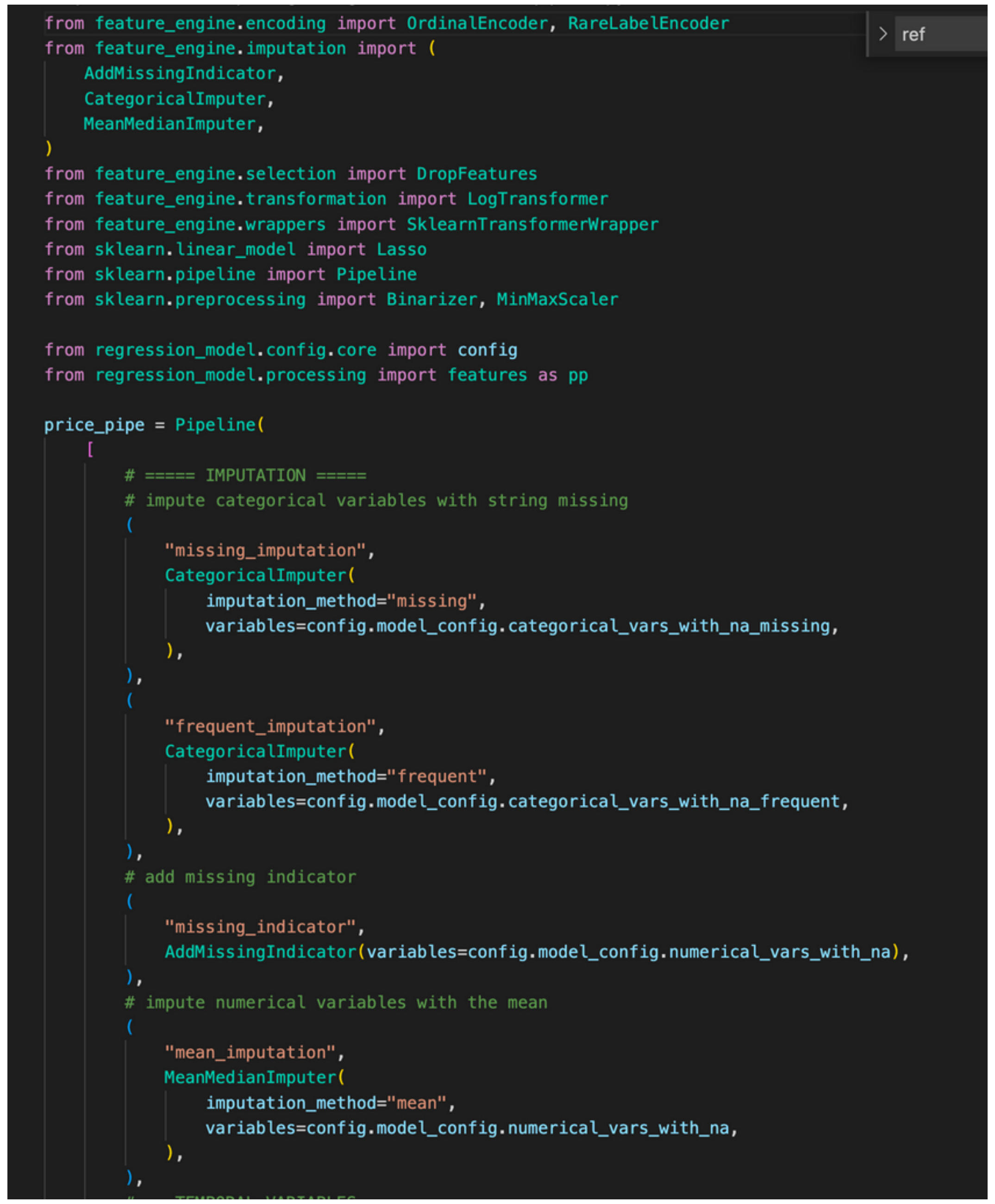


```python
import numpy as np
from config.core import config
from pipeline import price_pipe
from processing.data_manager import load_dataset, save_pipeline
from sklearn.model_selection import train_test_split


def run_training() -> None:
    """Train the model."""

    # read training data
    data = load_dataset(file_name=config.app_config.training_data_file)

    # divide train and test
    X_train, X_test, y_train, y_test = train_test_split(
        data[config.model_config.features],  # predictors
        data[config.model_config.target],
        test_size=config.model_config.test_size,
        # we are setting the random seed here
        # for reproducibility
        random_state=config.model_config.random_state,
    )
    y_train = np.log(y_train)

    # fit model
    price_pipe.fit(X_train, y_train)

    # persist trained model
    save_pipeline(pipeline_to_persist=price_pipe)


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

1. There is one function run_training, where we are calling the utility function load_dataset that come data_manager module.
2. From the pipeline module we using price_pipe for feature engineering and regression modeling
3. save_pipeline is used to persist the model. save_pipeline come from data_manager where details are mention there.
4. In the main we are calling run_training()

StatusNeo

Pipeline module

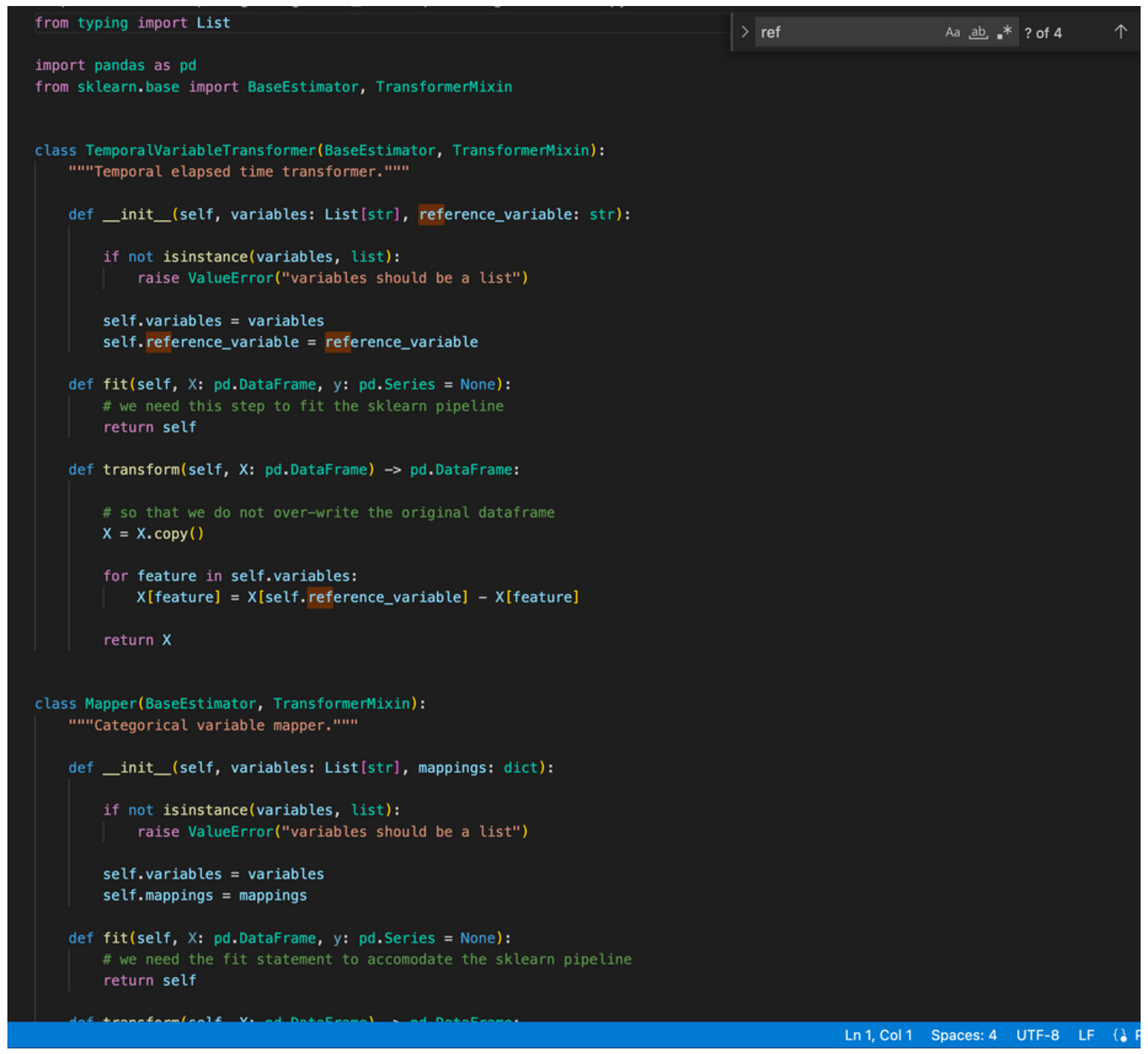


```
from feature_engine.encoding import OrdinalEncoder, RareLabelEncoder
from feature_engine.imputation import (
    AddMissingIndicator,
    CategoricalImputer,
    MeanMedianImputer,
)
from feature_engine.selection import DropFeatures
from feature_engine.transformation import LogTransformer
from feature_engine.wrappers import SklearnTransformerWrapper
from sklearn.linear_model import Lasso
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import Binarizer, MinMaxScaler

from regression_model.config.core import config
from regression_model.processing import features as pp

price_pipe = Pipeline(
    [
        # ===== IMPUTATION =====
        # impute categorical variables with string missing
        (
            "missing_imputation",
            CategoricalImputer(
                imputation_method="missing",
                variables=config.model_config.categorical_vars_with_na_missing,
            ),
        ),
        (
            "frequent_imputation",
            CategoricalImputer(
                imputation_method="frequent",
                variables=config.model_config.categorical_vars_with_na_frequent,
            ),
        ),
        # add missing indicator
        (
            "missing_indicator",
            AddMissingIndicator(variables=config.model_config.numerical_vars_with_na),
        ),
        # impute numerical variables with the mean
        (
            "mean_imputation",
            MeanMedianImputer(
                imputation_method="mean",
                variables=config.model_config.numerical_vars_with_na,
            ),
        ),
```

1. Pipeline module code similar to the pipeline we created in the jupyter notebook.
2. config comes from config module and config file.
   a. Temporal variable comes from features module
   b. Mapper comes from features module

StatusNeo

Feature engineering module



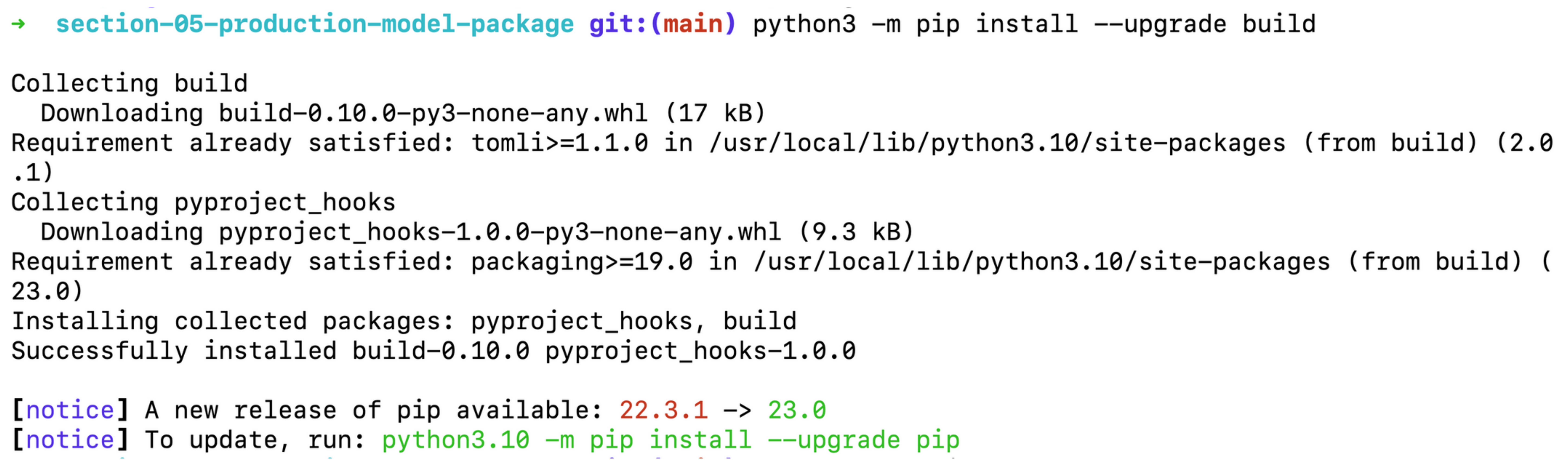
```python
from typing import List

import pandas as pd
from sklearn.base import BaseEstimator, TransformerMixin


class TemporalVariableTransformer(BaseEstimator, TransformerMixin):
    """Temporal elapsed time transformer."""

    def __init__(self, variables: List[str], reference_variable: str):

        if not isinstance(variables, list):
            raise ValueError("variables should be a list")

        self.variables = variables
        self.reference_variable = reference_variable

    def fit(self, X: pd.DataFrame, y: pd.Series = None):
        # we need this step to fit the sklearn pipeline
        return self

    def transform(self, X: pd.DataFrame) -> pd.DataFrame:

        # so that we do not over-write the original dataframe
        X = X.copy()

        for feature in self.variables:
            X[feature] = X[self.reference_variable] - X[feature]

        return X


class Mapper(BaseEstimator, TransformerMixin):
    """Categorical variable mapper."""

    def __init__(self, variables: List[str], mappings: dict):

        if not isinstance(variables, list):
            raise ValueError("variables should be a list")

        self.variables = variables
        self.mappings = mappings

    def fit(self, X: pd.DataFrame, y: pd.Series = None):
        # we need the fit statement to accomodate the sklearn pipeline
        return self
```

TemporalVariableTransformer -> Used for transformation of Elasped Time

Mapper Class -> This is used for mapping of categorical variable

StatusNeo

Packaging



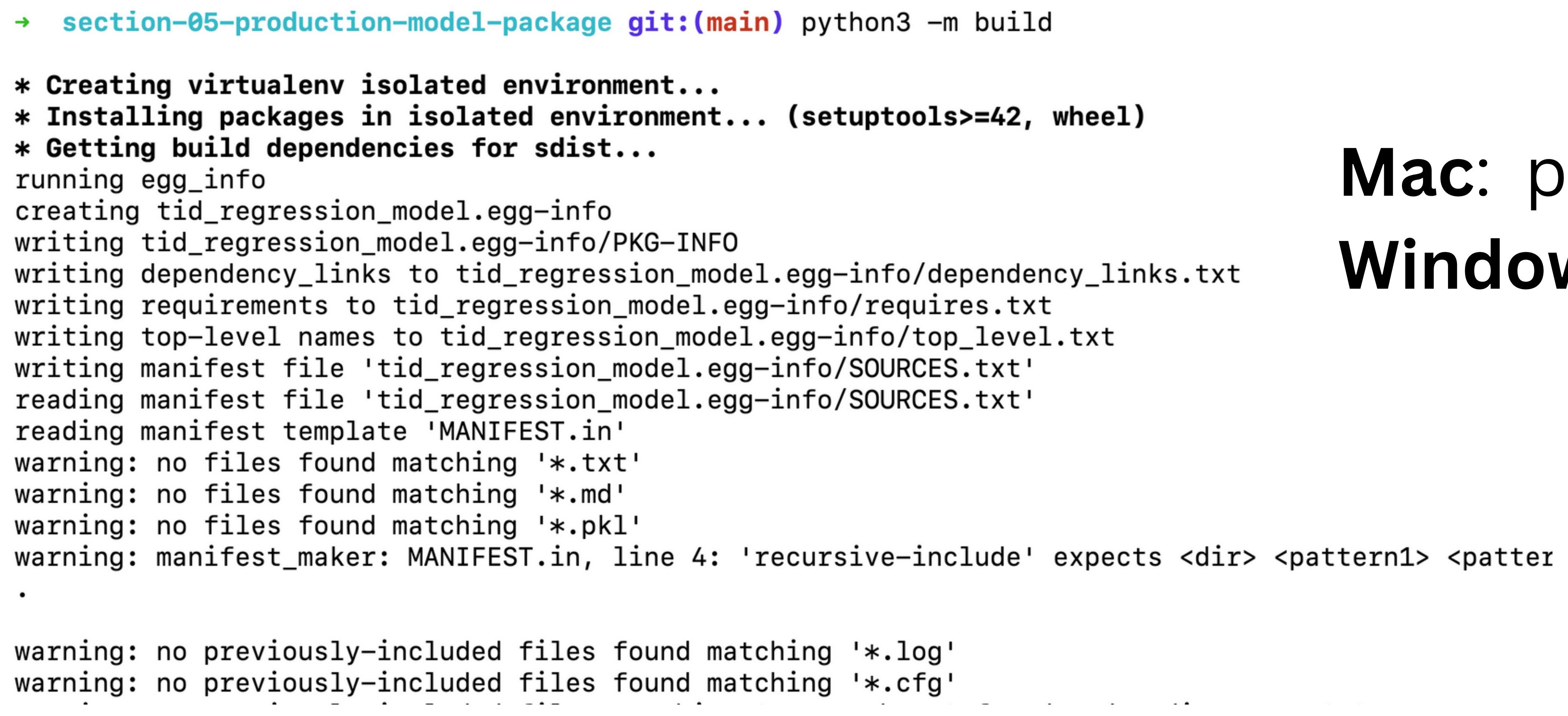
```
➜  section-05-production-model-package git:(main) python3 -m pip install --upgrade build

Collecting build
  Downloading build-0.10.0-py3-none-any.whl (17 kB)
Requirement already satisfied: tomli>=1.1.0 in /usr/local/lib/python3.10/site-packages (from build) (2.0
.1)
Collecting pyproject_hooks
  Downloading pyproject_hooks-1.0.0-py3-none-any.whl (9.3 kB)
Requirement already satisfied: packaging>=19.0 in /usr/local/lib/python3.10/site-packages (from build) (
23.0)
Installing collected packages: pyproject_hooks, build
Successfully installed build-0.10.0 pyproject_hooks-1.0.0

[notice] A new release of pip available: 22.3.1 -> 23.0
[notice] To update, run: python3.10 -m pip install --upgrade pip
```

Mac: python3 -m pip install --upgrade build

Windows: py -m pip install --upgrade build

StatusNeo

Packaging

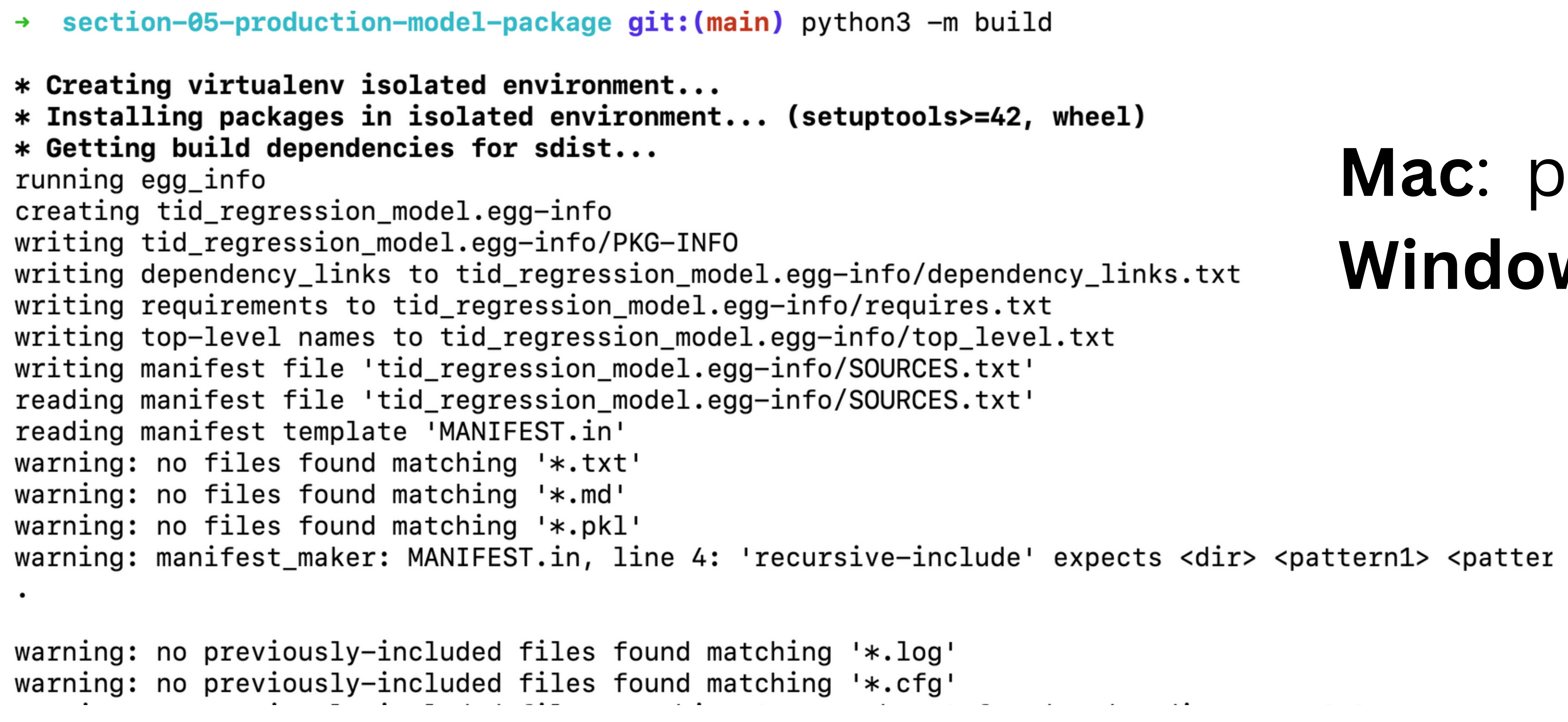


```
➜  section-05-production-model-package git:(main) python3 -m build

* Creating virtualenv isolated environment...
* Installing packages in isolated environment... (setuptools>=42, wheel)
* Getting build dependencies for sdist...
running egg_info
creating tid_regression_model.egg-info
writing tid_regression_model.egg-info/PKG-INFO
writing dependency_links to tid_regression_model.egg-info/dependency_links.txt
writing requirements to tid_regression_model.egg-info/requires.txt
writing top-level names to tid_regression_model.egg-info/top_level.txt
writing manifest file 'tid_regression_model.egg-info/SOURCES.txt'
reading manifest file 'tid_regression_model.egg-info/SOURCES.txt'
reading manifest template 'MANIFEST.in'
warning: no files found matching '*.txt'
warning: no files found matching '*.md'
warning: no files found matching '*.pkl'
warning: manifest_maker: MANIFEST.in, line 4: 'recursive-include' expects <dir> <pattern1> <patter
.

warning: no previously-included files found matching '*.log'
warning: no previously-included files found matching '*.cfg'
```

Mac: python3 -m build

Windows: py -m build

Packaging



```
➜  section-05-production-model-package git:(main) python3 -m build

* Creating virtualenv isolated environment...
* Installing packages in isolated environment... (setuptools>=42, wheel)
* Getting build dependencies for sdist...
running egg_info
creating tid_regression_model.egg-info
writing tid_regression_model.egg-info/PKG-INFO
writing dependency_links to tid_regression_model.egg-info/dependency_links.txt
writing requirements to tid_regression_model.egg-info/requires.txt
writing top-level names to tid_regression_model.egg-info/top_level.txt
writing manifest file 'tid_regression_model.egg-info/SOURCES.txt'
reading manifest file 'tid_regression_model.egg-info/SOURCES.txt'
reading manifest template 'MANIFEST.in'
warning: no files found matching '*.txt'
warning: no files found matching '*.md'
warning: no files found matching '*.pkl'
warning: manifest_maker: MANIFEST.in, line 4: 'recursive-include' expects <dir> <pattern1> <patter
.

warning: no previously-included files found matching '*.log'
warning: no previously-included files found matching '*.cfg'
```

Mac: python3 -m build

Windows: py -m build

StatusNeo

Assignment 2

Build a Production Code for the Titatnic Pipeline Created in Assignment 1

Timeline of the Assignment : Tuesday 21st Febraury

Flow to follow:

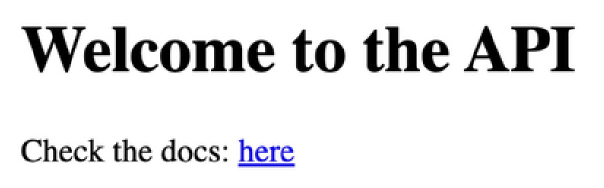
1. Prepare your model in the Jupyter Notebook
2. Copy the example repo that we did in class
3. Update the config file as per the titanic model
4. Load your new data
5. Start with a simple pipeline
6. Create the train, test , predict module
7. Create the predict tests and get them running
8. Update the pipeline if needed
9. Update the test
10. Add the rest of the package code (including requirements.txt)

Submit your code to your github repo

StatusNeo

Build an API Locally

1. Clone the model serving API from Git
2. Run the tox -e run (command) -> This will setup the tox environment
3. Now lets go to the http://localhost:8001/



Welcome to the API

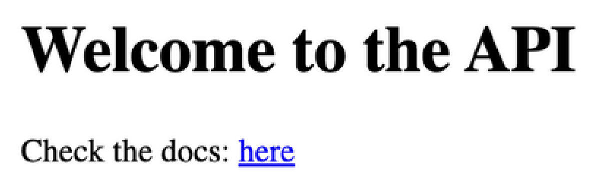
Check the docs: here

4. Now Lets try

StatusNeo

Build an API Locally

1. Clone the model serving API from Git
2. Run the tox -e run (command) -> This will setup the tox environment
3. Now lets go to the http://localhost:8001/



Welcome to the API

Check the docs: here

4. Now Lets try