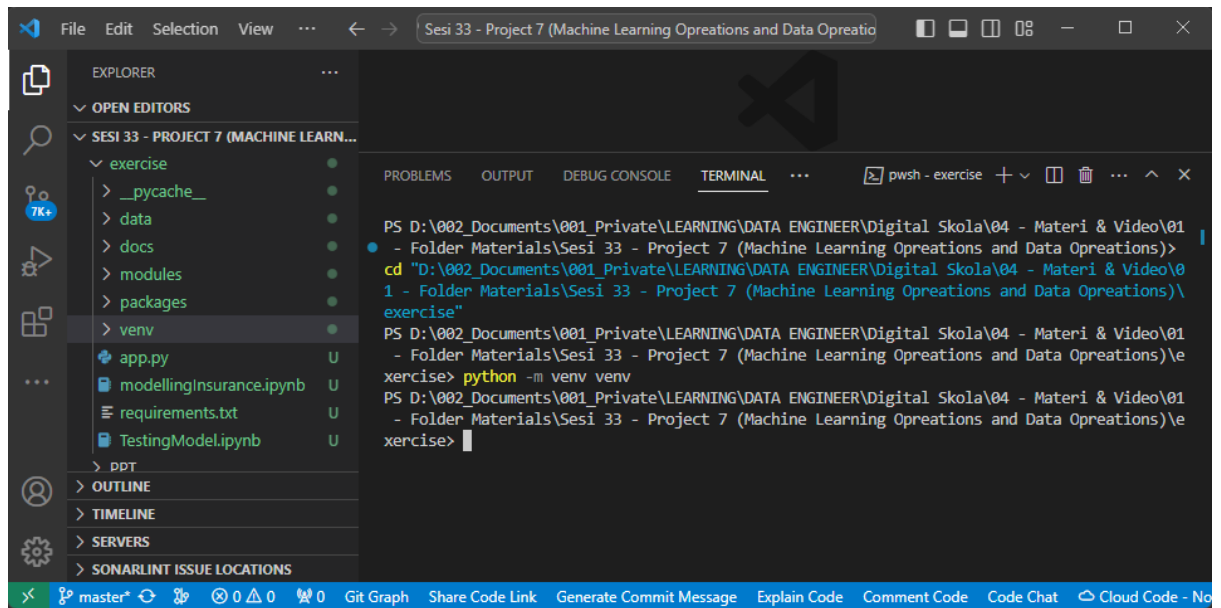


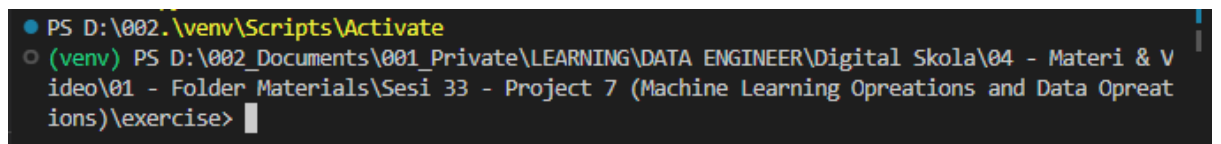
1. Create venv di powershell `python -m venv venv`



The screenshot shows the Visual Studio Code interface. The Explorer pane on the left shows a project structure with a folder named 'exercise' containing files like 'app.py', 'modellingInsurance.ipynb', 'requirements.txt', and 'TestingModel.ipynb'. The Terminal pane on the right shows the following PowerShell commands and output:

```
PS D:\002_Documents\001_Private\LEARNING\DATA ENGINEER\Digital Skola\04 - Materi & Video\01 - Folder Materials\Sesi 33 - Project 7 (Machine Learning Opreations and Data Opreations)> cd "D:\002_Documents\001_Private\LEARNING\DATA ENGINEER\Digital Skola\04 - Materi & Video\01 - Folder Materials\Sesi 33 - Project 7 (Machine Learning Opreations and Data Opreations)\exercise"
PS D:\002_Documents\001_Private\LEARNING\DATA ENGINEER\Digital Skola\04 - Materi & Video\01 - Folder Materials\Sesi 33 - Project 7 (Machine Learning Opreations and Data Opreations)\exercise> python -m venv venv
PS D:\002_Documents\001_Private\LEARNING\DATA ENGINEER\Digital Skola\04 - Materi & Video\01 - Folder Materials\Sesi 33 - Project 7 (Machine Learning Opreations and Data Opreations)\exercise>
```

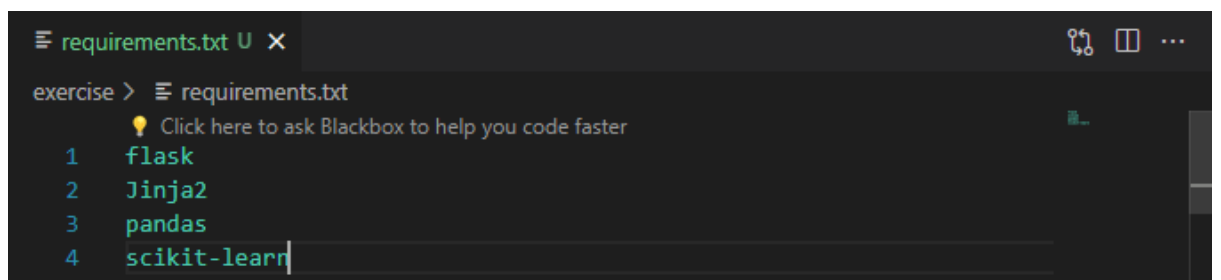
2. Activate venv



The screenshot shows the Visual Studio Code terminal with the following PowerShell commands and output:

```
PS D:\002...\venv\Scripts\Activate
(venv) PS D:\002_Documents\001_Private\LEARNING\DATA ENGINEER\Digital Skola\04 - Materi & Video\01 - Folder Materials\Sesi 33 - Project 7 (Machine Learning Opreations and Data Opreations)\exercise>
```

3. Create requirements.txt



The screenshot shows the Visual Studio Code editor with a file named 'requirements.txt' open. The file contains the following text:

```
1 flask
2 Jinja2
3 pandas
4 scikit-learn
```

4. Install library-library yang (pip install -r requirements.txt) dibutuhkan dari requirements.txt

The screenshot shows the Visual Studio Code interface with a project named 'Sesi 33 - Project 7 (Machine Learning Operations and Data Operations)'. The Explorer panel on the left shows the file structure, including a folder 'exercise' containing files like 'requirements.txt', 'app.py', 'modellingInsurance.ipynb', and 'TestingModel.ipynb'. The main editor displays the 'requirements.txt' file with the following content:

```
1 flask
2 Jinja2
3 pandas
4 scikit-learn
```

The TERMINAL panel at the bottom shows the output of the command 'pip install -r requirements.txt'. The output indicates that several dependencies are already satisfied, while others are being installed or updated. The dependencies listed include flask, Jinja2, pandas, scikit-learn, Werkzeug, itsdangerous, click, blinker, MarkupSafe, numpy, and python-dateutil.

5. Create / running script app.py :

```
# Import Modul dan Class
from flask import Flask
from flask import request
from flask import jsonify
import pandas as pd
from modules.insurance_model import InsuranceModel

# Inisialisasi Flask App
app = Flask(__name__)
```

```

# Route untuk Halaman Utama
@app.route('/')
def home():
    return "Welcome to the API modelling Prediction Insurance!" # Respons
    untuk halaman utama

# Route untuk Predict
@app.route('/predict', methods=['POST'])
def predict():
    # Mengambil data JSON dari permintaan
    data = request.get_json()

    # Membuat DataFrame pandas dari data JSON yang diterima
    df = pd.DataFrame(data)

    # Melakukan prediksi menggunakan metode runModel dari kelas
    InsuranceModel
    result_predict = InsuranceModel().runModel(df, typed='single')

    # Mengembalikan hasil prediksi dalam format JSON dengan status
    "predicted"
    return jsonify({
        "status": "predicted",
        "predicted_results": result_predict
    })

# Running Flask App
if __name__ == "__main__":
    app.run(port=8000) # Running aplikasi Flask pada port 8000

```

6. Create /running script `insurance_model.py` untuk memuat model predict:

```

from modules.insurance_pre import InsurancePre # Mengimpor modul
InsurancePre dari paket modules

import os # Mengimpor modul os untuk interaksi dengan sistem operasi
import pickle # Mengimpor modul pickle untuk menyimpan dan memuat objek
Python
import time # Mengimpor modul time untuk manajemen waktu

import pandas as pd # Mengimpor modul pandas dengan alias pd
import numpy as np # Mengimpor modul numpy dengan alias np

import warnings # Mengimpor modul warnings untuk mengatur peringatan
warnings.filterwarnings('ignore') # Menonaktifkan peringatan

class InsuranceModel(): # Mendefinisikan kelas InsuranceModel
    def __init__(self): # Metode inisialisasi kelas
        pass # Tidak melakukan apa-apa

    def runModel(self, data, typed='multi'): # Metode untuk menjalankan
    model prediksi asuransi
        # path = os.getcwd()+"/"+"packages"+"/"
        path = os.getcwd()+"/modules/packages/" # Mendefinisikan path ke
    direktori yang berisi model dan praproses

        # Memuat model dan praproses yang diperlukan dari file yang
    disimpan menggunakan pickle
        model = pickle.load(open(path +
    'model_InsuranceRecommendation.pkl', 'rb'))

```

```

col_p = pickle.load(open(path + 'columnPreparation.pkl', 'rb'))
col_m = pickle.load(open(path + 'columnModelling.pkl', 'rb'))

X = data[col_p] # Memilih kolom yang diperlukan dari data
colEncoder, colpOneHotEncoder, colStandarScaler =
InsurancePre().colPreparation() # Memanggil fungsi colPreparation dari
modul InsurancePre

# Melakukan praproses pada data sesuai dengan praproses yang telah
ditentukan sebelumnya
for col in X.columns:
    prep = pickle.load(open(path + 'prep' + col + '.pkl', 'rb')) #
Memuat praproses dari file yang disimpan menggunakan pickle
    if col in colpOneHotEncoder:
        dfTemp = pd.DataFrame(prepare.transform(X[[col]]).toarray())
# Transformasi menggunakan praproses yang sesuai
        X = pd.concat([X.drop(col, axis=1), dfTemp], axis=1)
    else:
        dfTemp = pd.DataFrame(prepare.transform(X[[col]]))
        X = pd.concat([X.drop(col, axis=1), dfTemp], axis=1)
X.columns = col_m # Menetapkan nama kolom yang baru

if typed == 'multi': # Jika jenis prediksi adalah multi-kelas
    y = model.predict(X) # Melakukan prediksi menggunakan model
yang telah dimuat
    return y # Mengembalikan hasil prediksi

elif typed == 'single': # Jika jenis prediksi adalah biner
    y = model.predict(X)[0] # Melakukan prediksi menggunakan model
yang telah dimuat
    if y == 0:
        return 0
    else:
        return 1
else:
    return False # Mengembalikan False jika jenis prediksi tidak
valid

```

7. Create / running script `insurance_pre.py` untuk menyiapkan kolom-kolom yang akan diproses sebelum melakukan praproses :

```

class InsurancePre(): # Mendefinisikan kelas InsurancePre
    def __init__(self): # Metode inisialisasi kelas
        pass # Tidak melakukan apa-apa

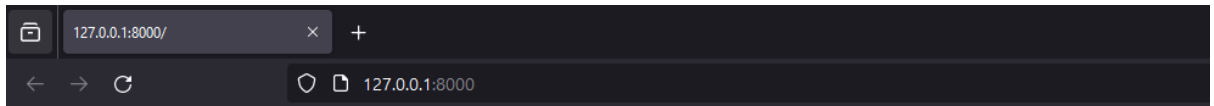
    def colPreparation(self): # Metode untuk persiapan kolom
        labelEncoder =
['Gender', 'Driving_License', 'Previously_Insured', 'Vehicle_Damage'] #
Daftar kolom untuk LabelEncoder
        oneHotEncoder =
['Vehicle_Age', 'Region_Code', 'Policy_Sales_Channel'] # Daftar kolom untuk
OneHotEncoder
        scalingStandar = ['Age', 'Annual_Premium', 'Vintage'] # Daftar
kolom untuk StandardScaler

        return labelEncoder, oneHotEncoder, scalingStandar #
Mengembalikan daftar kolom untuk persiapan praproses

```

8. Running flask run -port 8000

```
(venv) PS D:\002_Documents\001_Private\LEARNING\DATA ENGINEER\Digital Skola\04 - Materi & Video\01 - Folder Materials\Sesi 33 - Project 7 (Machine Learning Opreations and Data Opreations)\exercise> flask run --port 8000
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:8000
Press CTRL+C to quit
```



Welcome to the API modelling Prediction Insurance!

9. Running endpoint <http://localhost:8000/predict> di Postman

