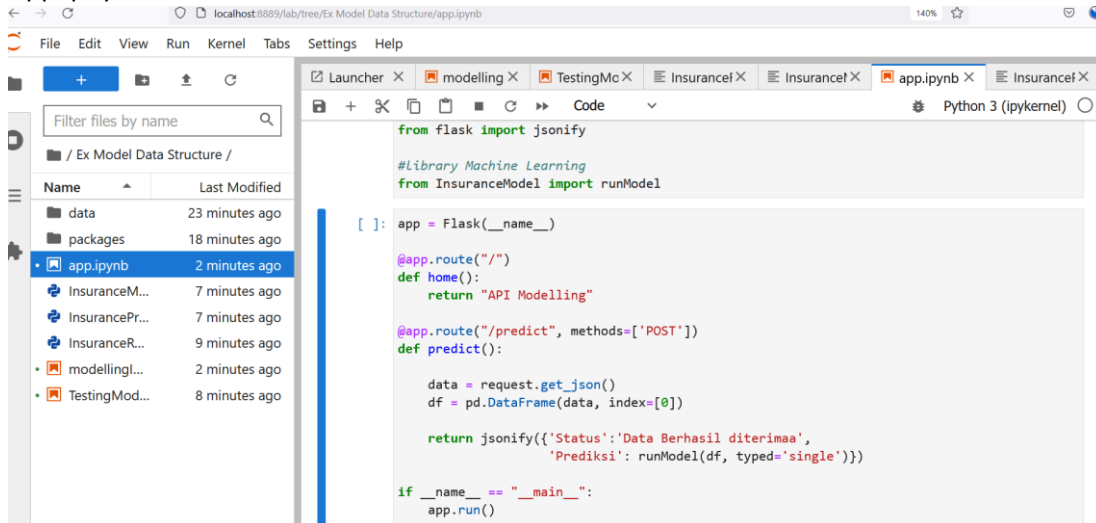


PROJECT 7 – WINDA AT

1. App.ipynb



The screenshot shows the JupyterLab interface with the file explorer on the left displaying the 'Ex Model Data Structure' directory. The file 'app.ipynb' is selected. The main editor shows the code for the 'app.ipynb' file, which is a Flask application. The code imports 'jsonify' from 'flask' and 'runModel' from 'InsuranceModel'. It defines a Flask app, a home route, and a predict route that calls 'runModel' with the request data.

```
from flask import jsonify

#Library Machine Learning
from InsuranceModel import runModel

[ ]: app = Flask(__name__)

@app.route("/")
def home():
    return "API Modelling"

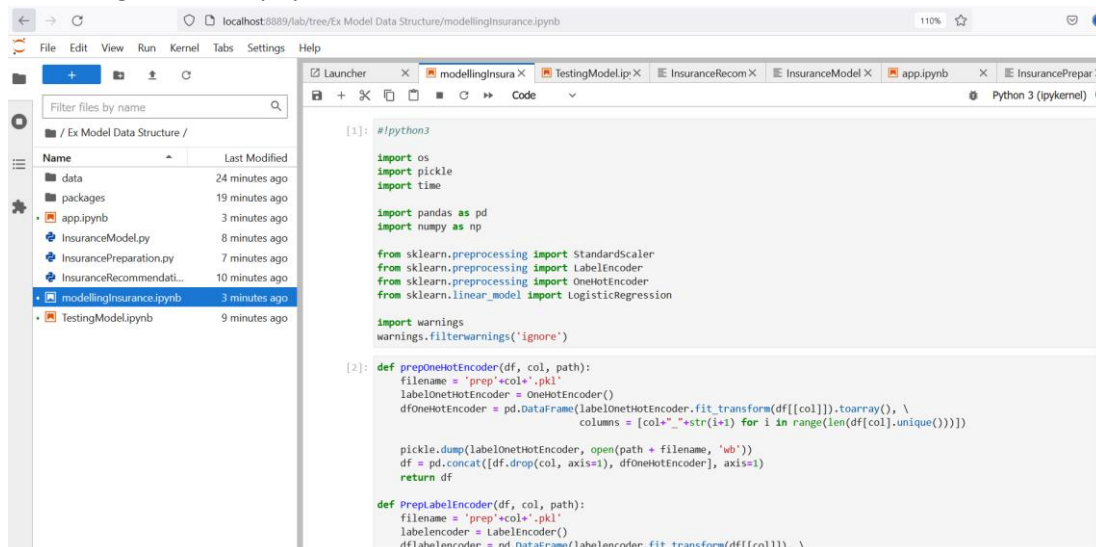
@app.route("/predict", methods=['POST'])
def predict():

    data = request.get_json()
    df = pd.DataFrame(data, index=[0])

    return jsonify({'Status': 'Data Berhasil diterima',
                    'Prediksi': runModel(df, typed='single')})

if __name__ == "__main__":
    app.run()
```

2. ModellingInsurance.ipynb



The screenshot shows the JupyterLab interface with the file explorer on the left displaying the 'Ex Model Data Structure' directory. The file 'ModellingInsurance.ipynb' is selected. The main editor shows the code for the 'ModellingInsurance.ipynb' file, which is a Python script. The code imports various libraries including 'os', 'pickle', 'time', 'pandas', 'numpy', 'sklearn', and 'warnings'. It defines two functions: 'prepOneHotEncoder' and 'PreLabelEncoder', which perform data preprocessing and encoding.

```
[1]: #!python3

import os
import pickle
import time

import pandas as pd
import numpy as np

from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.linear_model import LogisticRegression

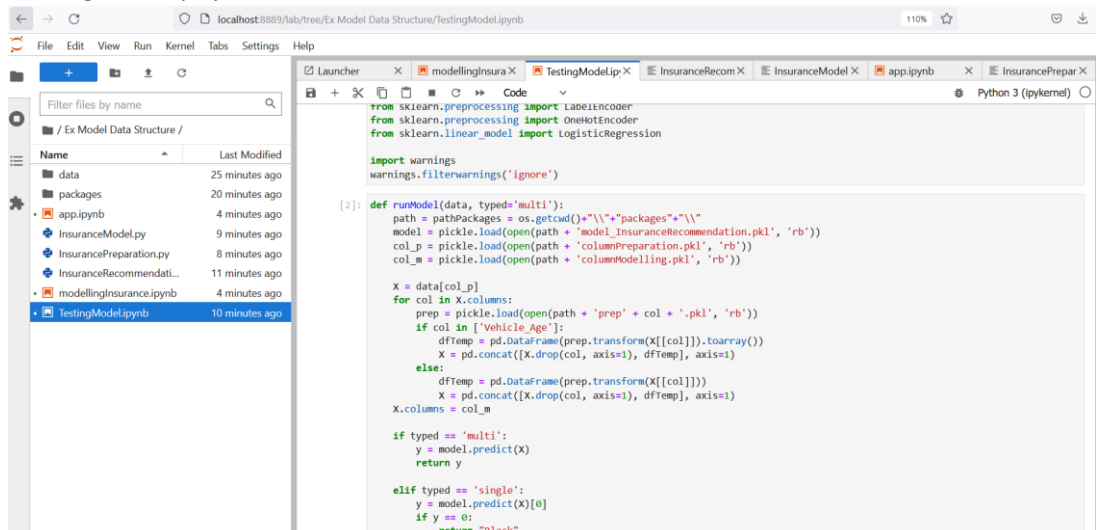
import warnings
warnings.filterwarnings('ignore')

[2]: def prepOneHotEncoder(df, col, path):
    filename = 'prep'+col+'.pkl'
    labelOneHotEncoder = OneHotEncoder()
    dfOneHotEncoder = pd.DataFrame(labelOneHotEncoder.fit_transform(df[[col]]).toarray(), \
                                  columns = [col+"_"+str(i+1) for i in range(len(df[col].unique()))])

    pickle.dump(labelOneHotEncoder, open(path + filename, 'wb'))
    df = pd.concat([df.drop(col, axis=1), dfOneHotEncoder], axis=1)
    return df

def PreLabelEncoder(df, col, path):
    filename = 'prep'+col+'.pkl'
    labelEncoder = LabelEncoder()
    dfLabelEncoder = pd.DataFrame(labelEncoder.fit_transform(df[[col]]), \
```

3. TestingMode.ipynb



The screenshot shows the JupyterLab interface with the file explorer on the left displaying the 'Ex Model Data Structure' directory. The file 'TestingMode.ipynb' is selected. The main editor shows the code for the 'TestingMode.ipynb' file, which is a Python script. The code imports various libraries including 'sklearn', 'pandas', 'numpy', 'os', 'pickle', and 'warnings'. It defines a function 'runModel' that performs data preprocessing and model prediction.

```
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.linear_model import LogisticRegression

import warnings
warnings.filterwarnings('ignore')

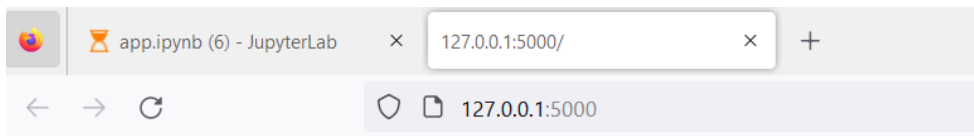
[2]: def runModel(data, typed='multi'):
    path = pathPackages + os.getcwd()+"\\*\\*\\*packages*\\*\\*
    model = pickle.load(open(path + 'model_InsuranceRecommendation.pkl', 'rb'))
    col_p = pickle.load(open(path + 'columnPreparation.pkl', 'rb'))
    col_m = pickle.load(open(path + 'columnModelling.pkl', 'rb'))

    X = data[col_p]
    for col in X.columns:
        prep = pickle.load(open(path + 'prep' + col + '.pkl', 'rb'))
        if col in ['Vehicle_Age']:
            dftemp = pd.DataFrame(prep.transform(X[[col]]).toarray())
            X = pd.concat([X.drop(col, axis=1), dftemp], axis=1)
        else:
            dftemp = pd.DataFrame(prep.transform(X[[col]]))
            X = pd.concat([X.drop(col, axis=1), dftemp], axis=1)
    X.columns = col_m

    if typed == 'multi':
        y = model.predict(X)
        return y

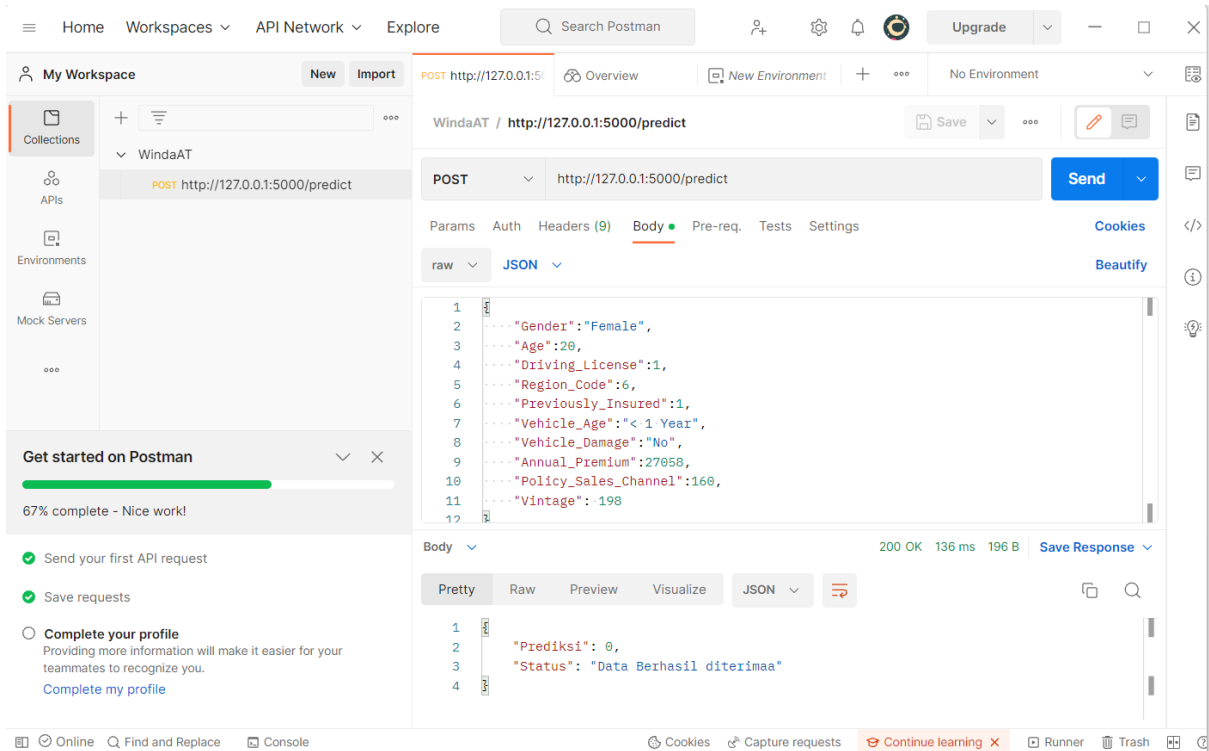
    elif typed == 'single':
        y = model.predict(X)[0]
        if y == 0:
            return "Black"
```

4. API Model



API Modelling

5. API



6. Anaconda Prompt

