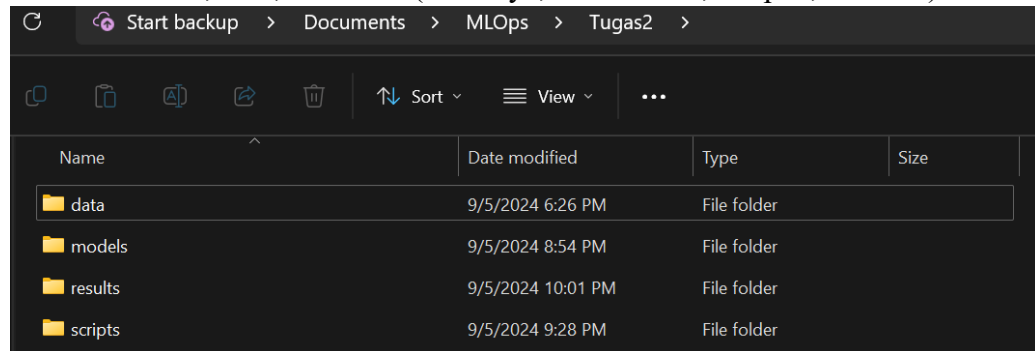
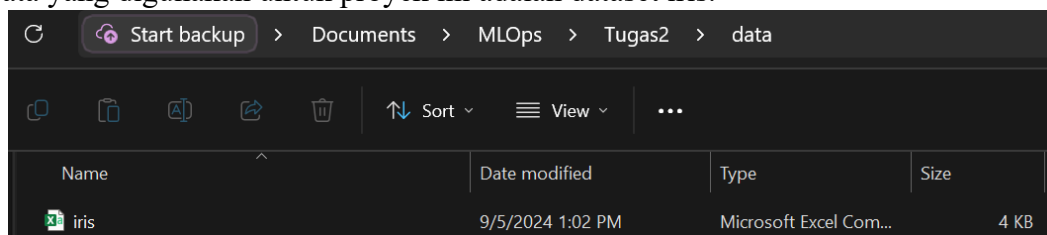


A. Menyiapkan Lingkungan Kerja:

Buat direktori proyek baru di komputer lokal. Lalu, siapkan struktur folder untuk memisahkan kode, data, dan hasil (misalnya, folder data/, scripts/, models/).



Data yang digunakan untuk proyek ini adalah dataset iris.



B. Menulis Skrip Python

Buat skrip Python untuk setiap tahap dalam siklus machine learning: data_prep.py, train_model.py, evaluate_model.py, deploy_model.py.

1. data_prep.py

```
data_prep.py
Tugas2 > scripts > data_prep.py > ...
...
1 import pandas as pd
2 from sklearn.preprocessing import StandardScaler
3
4 def load_and_clean_data(file_path):
5     df = pd.read_csv(file_path)
6     df.drop_duplicates(inplace=True)
7     return df
8
9 def handle_outliers(df):
10     numerical_columns = df.select_dtypes(include=['float64', 'int64'])
11
12     Q1 = numerical_columns.quantile(0.25)
13     Q3 = numerical_columns.quantile(0.75)
14     IQR = Q3 - Q1
15
16     lower_limit = Q1['sepal.width'] - 1.5 * IQR['sepal.width']
17     upper_limit = Q3['sepal.width'] + 1.5 * IQR['sepal.width']
18
19     df_cleaned = df[(df['sepal.width'] >= lower_limit) & (df['sepal.width'] <= upper_limit)]
20     return df_cleaned
21
22 def normalize_data(df_cleaned):
23     variety = df_cleaned['variety']
24     numerical_columns_cleaned = df_cleaned.drop(columns=['variety'])
25
26     scaler = StandardScaler()
27     numerical_columns_normalized = pd.DataFrame(scaler.fit_transform(numerical_columns_cleaned), columns=numerical_columns_cleaned.columns)
28
29     df_normalized = numerical_columns_normalized.copy()
30     df_normalized['variety'] = variety.reset_index(drop=True)
31     return df_normalized
32
33 def preprocess_iris_data(file_path, output_file_path):
34     df = load_and_clean_data(file_path)
35     df_cleaned = handle_outliers(df)
36     df_normalized = normalize_data(df_cleaned)
37
38     df_normalized.to_csv(output_file_path, index=False)
39     return df_normalized
40
41 if __name__ == "__main__":
42     input_file = 'data/iris.csv'
43     output_file = 'data/processed_iris.csv'
44     df_prepared = preprocess_iris_data(input_file, output_file)
```

Kode data_prep.py ini digunakan untuk *preprocessing* atau pembersihan data pada dataset Iris. Di dalam kode ini dilakukan, pembersihan data (drop duplicate data), penanganan *outlier*, dan normalisasi fitur menggunakan *standard scaler*. Hasil dari kode ini akan disimpan dalam file csv bernama 'processed_iris.csv'.

2. train_model.py

```
train_model.py
Tugas2 > scripts > train_model.py > ...
You, 48 minutes ago | 1 author (You)
1 import pandas as pd
2 from sklearn.model_selection import train_test_split, GridSearchCV
3 from sklearn.preprocessing import LabelEncoder
4 from sklearn.tree import DecisionTreeClassifier
5 import pickle
6
7 def load_data(file_path):
8     df = pd.read_csv(file_path)
9     X = df.drop(columns=['variety'])
10    y = df['variety']
11    le = LabelEncoder()
12    y_encoded = le.fit_transform(y)
13    return X, y_encoded
14
15 def train_model(X_train, y_train):
16     param_grid = {'max_depth': range(1, 11)}
17     dt = DecisionTreeClassifier(random_state=42)
18     grid_search = GridSearchCV(estimator=dt, param_grid=param_grid, cv=10, scoring='accuracy')
19     grid_search.fit(X_train, y_train)
20     best_max_depth = grid_search.best_params_['max_depth']
21     best_accuracy = grid_search.best_score_
22     print(f"Best max_depth: {best_max_depth}, Best cross-validated accuracy: {best_accuracy}")
23     model = DecisionTreeClassifier(max_depth=best_max_depth, random_state=42)
24     model.fit(X_train, y_train)
25     return model
26
27 def save_model(model, output_file):
28     with open(output_file, 'wb') as f:
29         pickle.dump(model, f)
30     print(f"Model saved to {output_file}")
31
32 if __name__ == "__main__":
33     input_file = 'data/processed_iris.csv'
34     model_output_file = 'models/decision_tree_model.pkl'
35     X, y = load_data(input_file)
36     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
37     model = train_model(X_train, y_train)
38     save_model(model, model_output_file)
```

Kode train_model.py melakukan pelatihan model *Decision Tree* pada dataset Iris yang sudah diproses sebelumnya. Kode ini menggunakan *GridSearchCV* untuk menemukan parameter (max_depth) terbaik, lalu model yang sudah dilatih disimpan ke dalam file bernama 'decision_tree_model.pkl' menggunakan *pickle*.

3. evaluate_model.py

```
evaluate_model.py
Tugas2 > scripts > evaluate_model.py > ...
You, 48 minutes ago | 1 author (You)

1 import pandas as pd
2 import pickle
3 from sklearn.metrics import accuracy_score, precision_score, f1_score, recall_score
4 from sklearn.model_selection import train_test_split
5 from sklearn.preprocessing import LabelEncoder
6 import json
7
8 def load_data(file_path):
9     df = pd.read_csv(file_path)
10    X = df.drop(columns=['variety'])
11    y = df['variety']
12    le = LabelEncoder()
13    y_encoded = le.fit_transform(y)
14    return X, y_encoded
15
16 def load_model(model_file):
17     with open(model_file, 'rb') as f:
18         model = pickle.load(f)
19     print(f"Model loaded from {model_file}")
20     return model
21
22 def evaluate_model(model, X_test, y_test):
23     y_pred = model.predict(X_test)
24     accuracy = accuracy_score(y_test, y_pred)
25     precision = precision_score(y_test, y_pred, average='weighted')
26     recall = recall_score(y_test, y_pred, average='weighted')
27     f1 = f1_score(y_test, y_pred, average='weighted')
28
29     evaluation_metrics = {
30         'accuracy': accuracy,
31         'precision': precision,
32         'recall': recall,
33         'f1_score': f1
34     }
35     return evaluation_metrics
36
37 def save_eval_metrics(metrics, output_file):
38     with open(output_file, 'w') as f:
39         json.dump(metrics, f, indent=4)
40     print(f"Evaluation metrics saved to {output_file}")
41
42 if __name__ == "__main__":
43     input_file = 'data/processed_iris.csv'
44     model_file = 'models/decision_tree_model.pkl'
45     output_file = 'results/evaluation_metrics.json'
46
47     X, y = load_data(input_file)
48     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
49     model = load_model(model_file)
50     evaluation_metrics = evaluate_model(model, X_test, y_test)
51     save_eval_metrics(evaluation_metrics, output_file)
52     print(evaluation_metrics)
```

Kode `evaluate_model.py` ini digunakan untuk memuat model Decision Tree yang sudah dilatih dan mengevaluasi performa model pada data pengujian menggunakan metrik evaluasi, yaitu *accuracy*, *precision*, *recall*, dan *f1-score*. Hasil evaluasi ini akan disimpan ke dalam file JSON bernama 'evaluation_metrics.json'.

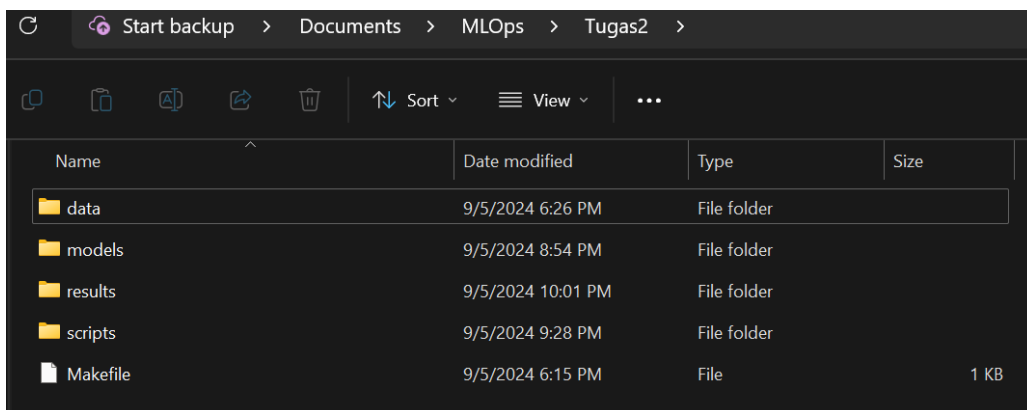
4. `deploy_model.py`

```
deploy_model.py
Tugas2 > scripts > deploy_model.py > ...
You, 1 hour ago | 1 author (You)
1 import pickle
2
3 def deploy_model(model_file, deploy_path):
4     with open(model_file, 'rb') as f:
5         model = pickle.load(f)
6
7     with open(deploy_path, 'wb') as f:
8         pickle.dump(model, f)
9
10 if __name__ == "__main__":
11     deploy_model('models/decision_tree_model.pkl', 'models/deployed_model.pkl')
```

Kode `deploy_model.py` digunakan untuk mendeploy model yang telah dilatih. Model yang disimpan sebelumnya dimuat dan disimpan kembali sebagai bentuk deployment dengan nama `'deployed_model.pkl'`.

C. Menyusun Makefile

Makefile merupakan file yang berguna untuk mengotomatisasi proses menjalankan berbagai tahapan dalam siklus hidup proyek *Machine Learning*.



Name	Date modified	Type	Size
data	9/5/2024 6:26 PM	File folder	
models	9/5/2024 8:54 PM	File folder	
results	9/5/2024 10:01 PM	File folder	
scripts	9/5/2024 9:28 PM	File folder	
Makefile	9/5/2024 6:15 PM	File	1 KB

Berikut merupakan isi file Makefile:

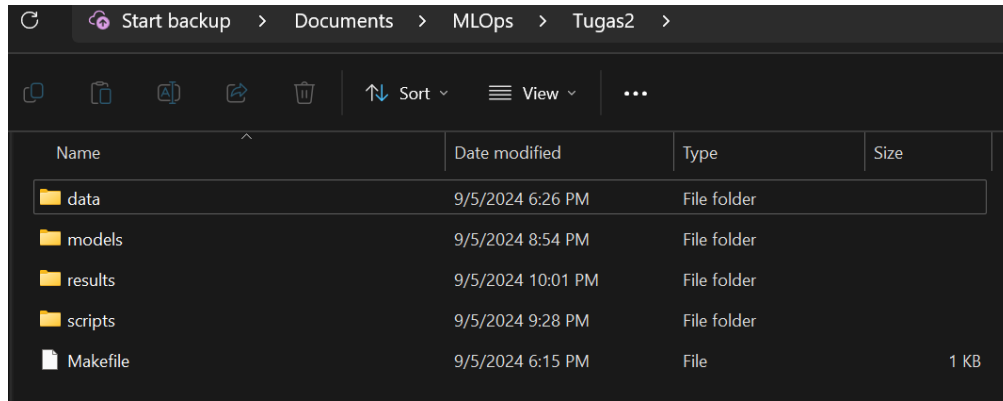
```
M Makefile
Tugas2 > M Makefile
You, 6 hours ago | 1 author (You)
1 DATA_DIR=data
2 SCRIPTS_DIR=scripts
3 MODELS_DIR=models
4 RESULTS_DIR=results
5
6 .PHONY: data train evaluate deploy clean
7
8 data:
9     python $(SCRIPTS_DIR)/data_prep.py
10
11 train:
12     python $(SCRIPTS_DIR)/train_model.py
13
14 evaluate:
15     python $(SCRIPTS_DIR)/evaluate_model.py
16
17 deploy:
18     python $(SCRIPTS_DIR)/deploy_model.py
19
20 clean:
21     rm -rf $(DATA_DIR)/.csv $(MODELS_DIR)/.pkl $(RESULTS_DIR)/*.txt
```

Terdapat beberapa perintah yang digunakan pada proyek ini, yaitu:

1. make data
Perintah ini berguna untuk mengumpulkan dan mempersiapkan data.
2. make train
Perintah ini berguna untuk melatih model menggunakan dataset yang sudah diproses.
3. make evaluate
Perintah ini berguna untuk mengevaluasi performa model.
4. make deploy
Perintah ini berguna untuk menyimpan model yang sudah dilatih atau membuat model siap digunakan di produksi.

Berikut merupakan output dari dijalankannya keempat perintah tersebut:

```
PS C:\Users\lyona\Documents\MLOps\Tugas2> make data
python scripts/data_prep.py
PS C:\Users\lyona\Documents\MLOps\Tugas2> make train
python scripts/train_model.py
Best max_depth: 3, Best cross-validated accuracy: 0.9818181818181818
Model saved to models/decision_tree_model.pkl
PS C:\Users\lyona\Documents\MLOps\Tugas2> make evaluate
python scripts/evaluate_model.py
Model loaded from models/decision_tree_model.pkl
Evaluation metrics saved to results/evaluation_metrics.json
{'accuracy': 0.9310344827586207, 'precision': 0.9416445623342174, 'recall': 0.9310344827586207, 'f1_score': 0.9295977011494252}
PS C:\Users\lyona\Documents\MLOps\Tugas2> make deploy
python scripts/deploy_model.py
PS C:\Users\lyona\Documents\MLOps\Tugas2>
```



D. Tantangan yang Dihadapi dan Solusi yang Diterapkan

Masalah yang dihadapi adalah tidak berjalannya perintah *make* di terminal. Solusi yang dilakukan adalah menginstal Chocolatey. Berikut merupakan cara penginstalan Chocolatey:

1. Masuk ke terminal atau Powershell sebagai admin (Run as administrator)
2. Jalankan perintah:

```
Set-ExecutionPolicy Bypass -Scope Process -Force;  
[System.Net.ServicePointManager]::SecurityProtocol =  
[System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object  
System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1'))
```

Setelah menginstal Chocolatey, lakukan perintah *choco install make* untuk menginstal *make*. Dengan dilakukan cara tersebut, perintah *make* sudah dapat dijalankan di terminal Visual Studio Code.

Referensi:

<https://earthly.dev/blog/makefiles-on-windows/>

E. Repository GitHub

Berikut merupakan link repository GitHub:

<https://github.com/lyonardgemilang/MLOps/tree/main/Tugas2>