

Tugas Pertemuan 5

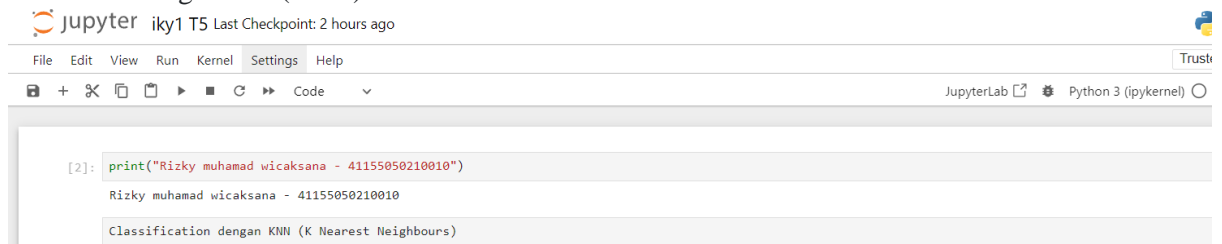
Nama : Rizky Muhamad Wicaksana

NPM : 41155050210010

Informatika A1

Machine Learning

1.0. K-Nearest Neighbours (KNN)



JupyterLab interface showing a code cell with the following code:

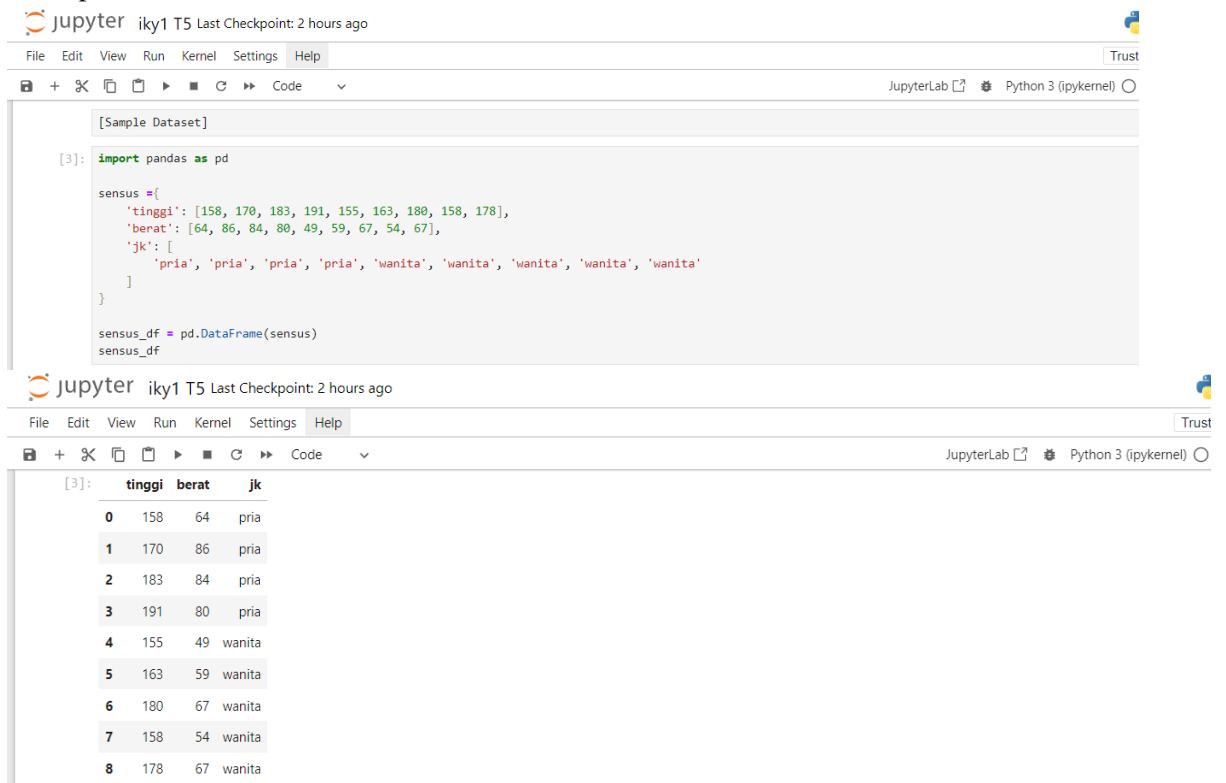
```
[2]: print("Rizky muhamad wicaksana - 41155050210010")
```

The output of the code cell is:

```
Rizky muhamad wicaksana - 41155050210010
```

Below the output, the text "Classification dengan KNN (K Nearest Neighbours)" is displayed.

1.1. Persiapan Data set



JupyterLab interface showing a code cell with the following code:

```
[3]: import pandas as pd

sensus = {
    'tinggi': [158, 170, 183, 191, 155, 163, 180, 158, 178],
    'berat': [64, 86, 84, 80, 49, 59, 67, 54, 67],
    'jk': [
        'pria', 'pria', 'pria', 'pria', 'wanita', 'wanita', 'wanita', 'wanita', 'wanita'
    ]
}

sensus_df = pd.DataFrame(sensus)
sensus_df
```

The output of the code cell is a DataFrame with the following data:

	tinggi	berat	jk
0	158	64	pria
1	170	86	pria
2	183	84	pria
3	191	80	pria
4	155	49	wanita
5	163	59	wanita
6	180	67	wanita
7	158	54	wanita
8	178	67	wanita

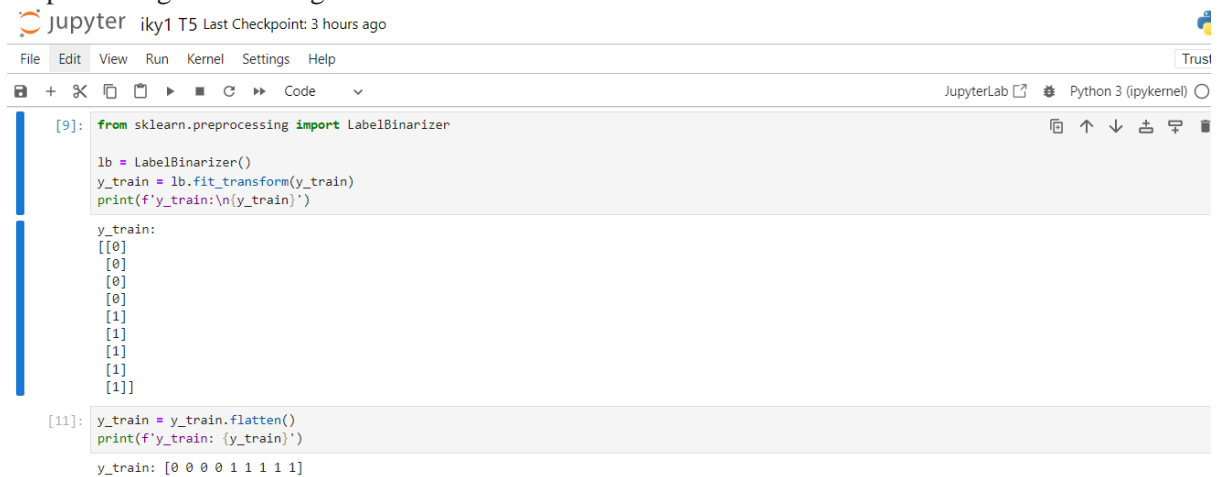
1.2. Visualisasi dataset



1.3. Pengantar classification dengan K-Nearest Neighbours | KNN



1.4. Preprocessing dataset dengan Label Binarizer



JupyterLab Python 3 (ipykernel)

```
[9]: from sklearn.preprocessing import LabelBinarizer


lb = LabelBinarizer()
y_train = lb.fit_transform(y_train)
print(f'y_train:\n{y_train}')
```

```
y_train:
[[0]
 [0]
 [0]
 [0]
 [1]
 [1]
 [1]
 [1]]
```

```
[11]: y_train = y_train.flatten()
print(f'y_train: {y_train}')
```

```
y_train: [0 0 0 0 1 1 1 1]
```

1.5. Training KNN Classification Model



JupyterLab Python 3 (ipykernel)

```
[18]: from sklearn.neighbors import KNeighborsClassifier

K = 3
model = KNeighborsClassifier(n_neighbors=K)
model.fit(X_train, y_train)
```

```
[18]: KNeighborsClassifier
KNeighborsClassifier(n_neighbors=3)
```

1.6. Prediksi dengan KNN Classification Model



JupyterLab Python 3 (ipykernel)

```
[15]: tinggi_badan = 155
berat_badan = 70
X_new = np.array([tinggi_badan, berat_badan]).reshape(1, -1)
X_new
```

```
[15]: array([[155, 70]])
```

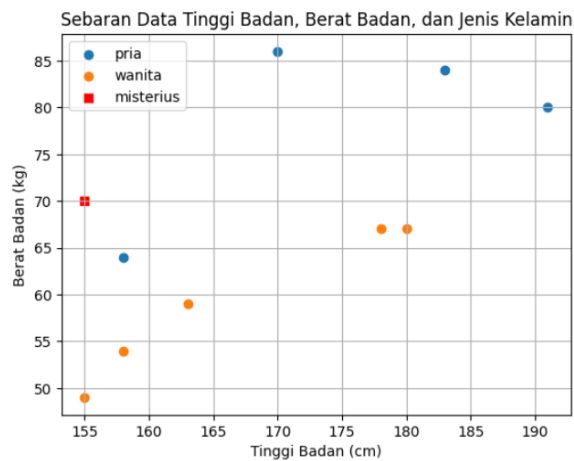
```
[16]: y_new = model.predict(X_new)
y_new
```

```
[16]: array([1])
```

```
[19]: lb.inverse_transform(y_new)
```

```
[19]: array(['wanita'], dtype='<U6')
```

1.7. Visualisasi Nearest Neighbours



1.8. Kalkulasi jarak dengan Euclidean Distance



JupyterLab Python 3 (ipykernel)

```
[25]: from scipy.spatial.distance import euclidean

data_jarak = [euclidean(misterius, d) for d in X_train]
data_jarak

[25]: [np.float64(6.708203932499369),
np.float64(21.93171219946131),
np.float64(31.304951684997057),
np.float64(37.36308338453881),
np.float64(21.0),
np.float64(13.601470508735444),
np.float64(25.179356624028344),
np.float64(16.278820596099706),
np.float64(23.194827009486403)]
```

JupyterLab Python 3 (ipykernel)

```
[26]: sensus_df['jarak'] = data_jarak
sensus_df.sort_values(['jarak'])

[26]:
```

	tinggi	berat	jk	jarak
0	158	64	pria	6.708204
5	163	59	wanita	13.601471
7	158	54	wanita	16.278821
4	155	49	wanita	21.000000
1	170	86	pria	21.931712
8	178	67	wanita	23.194827
6	180	67	wanita	25.179357
2	183	84	pria	31.304952
3	191	80	pria	37.363083

1.9. Evaluasi KNN Classification Model | Persiapan testing set

JupyterLab Python 3 (ipykernel)

```
[Evaluasi KNN Classification Model]

[Testing Set]

[27]: X_test = np.array([[168, 65], [180, 96], [160, 52], [169, 67]])
y_test = lb.transform(np.array(['pria', 'pria', 'wanita', 'wanita'])).flatten()

print(f'X_train:\n{X_test}\n')
print(f'y_test:\n{y_test}')

X_train:
[[168  65]
 [180  96]
 [160  52]
 [169  67]]

y_test:
[0 0 1 1]

[Prediksi terhadap testing set]

[28]: y_pred = model.predict(X_test)
y_pred

[28]: array([1, 0, 1, 1])
```

1.10. Evaluasi model dengan accuracy score

JupyterLab Python 3 (ipykernel)

```
[Evaluasi Accuracy]

[29]: from sklearn.metrics import accuracy_score

acc = accuracy_score(y_test, y_pred)

print(f'Accuracy: {acc}')

Accuracy: 0.75
```

1.11. Evaluasi model dengan precision score

JupyterLab Python 3 (ipykernel)

```
[Evaluasi Precision]

[30]: from sklearn.metrics import precision_score

      prec = precision_score(y_test, y_pred)

      print(f'Precision: {prec}')

      Precision: 0.6666666666666666
```

1.12. Evaluasi model dengan recall score

JupyterLab Python 3 (ipykernel)

```
[Evaluasi Recall]

[31]: from sklearn.metrics import recall_score

      rec = recall_score(y_test, y_pred)

      print(f'Recall: {rec}')

      Recall: 1.0
```

1.13. Evaluasi model dengan F1 score

JupyterLab Python 3 (ipykernel)

```
[Evaluasi F1 Score]

[33]: from sklearn.metrics import f1_score

      f1 = f1_score(y_test, y_pred)

      print(f'F1-score: {f1}')

      F1-score: 0.8
```

1.14. Evaluasi model dengan classification report

JupyterLab Python 3 (ipykernel)

```
[Classification Report]

[34]: from sklearn.metrics import classification_report

      cls_report = classification_report(y_test, y_pred)

      print(f'Classification Report:\n{cls_report}')

      Classification Report:
               precision    recall  f1-score   support

      0               1.00      0.50      0.67         2
      1               0.67      1.00      0.80         2

   accuracy              0.75         4
  macro avg              0.83         4
 weighted avg              0.75         4
```

1.15. Evaluasi model dengan Mathews Correlation Coefficient

JupyterLab Python 3 (ipykernel)

```
[Matthews Correlation Coefficient]

[35]: from sklearn.metrics import matthews_corrcoef

      mcc = matthews_corrcoef(y_test, y_pred)

      print(f'MCC: {mcc}')

      MCC: 0.5773502691896258
```

2.0. Support Vector Machine (SVM)

```
jupyter iky2 T5 Last Checkpoint: 18 hours ago
File Edit View Run Kernel Settings Help Trust
JupyterLab Python 3 (ipykernel)

[1]: print("Rizky muhamad wicaksana - 41155050210010")
Rizky muhamad wicaksana - 41155050210010

[Classification Task dengan Support Vector Machine (SVM)]
```

- 2.1. Pengenalan Decision Boundary & Hyperplane
- 2.2. Pengenalan Support Vector & Maximum Margin
- 2.3. Pengenalan kondisi Linearly Inseparable dan Kernel Tricks
- 2.4. Pengenalan MNIST Handwritten Digits Dataset

```
jupyter iky2 T5 Last Checkpoint: 19 hours ago
File Edit View Run Kernel Settings Help Trust
JupyterLab Python 3 (ipykernel)

[Dataset: The MNIST database of handwritten digits]

[18]: print("Rizky muhamad wicaksana - 41155050210010")
Rizky muhamad wicaksana - 41155050210010

[2]: from sklearn.datasets import fetch_openml

X, y = fetch_openml('mnist_784', data_home='./dataset/mnist', return_X_y=True)
X.shape

[2]: (70000, 784)

[8]: import matplotlib.pyplot as plt
import matplotlib.cm as cm

pos = 1
for data in X.to_numpy()[0:8]:
    plt.subplot(1, 8, pos)
    plt.imshow(data.reshape((28, 28)),
               cmap=cm.Greys_r)
    plt.axis('off')
    pos += 1

plt.show()

5 0 4 1 9 2 1 3
```

```
jupyter iky2 T5 Last Checkpoint: 19 hours ago
File Edit View Run Kernel Settings Help Trust
JupyterLab Python 3 (ipykernel)

[9]: y[:8]

[9]: 0 5
1 0
2 4
3 1
4 9
5 2
6 1
7 3
Name: class, dtype: category
Categories (10, object): ['0', '1', '2', '3', ..., '6', '7', '8', '9']

[12]: # X_train = X[:60000]
# y_train = y[:60000]
# X_test = X[60000:]
# y_test = y[60000:]

X_train = X[:1000]
y_train = y[:1000]
X_test = X[69000:]
y_test = y[69000:]
```

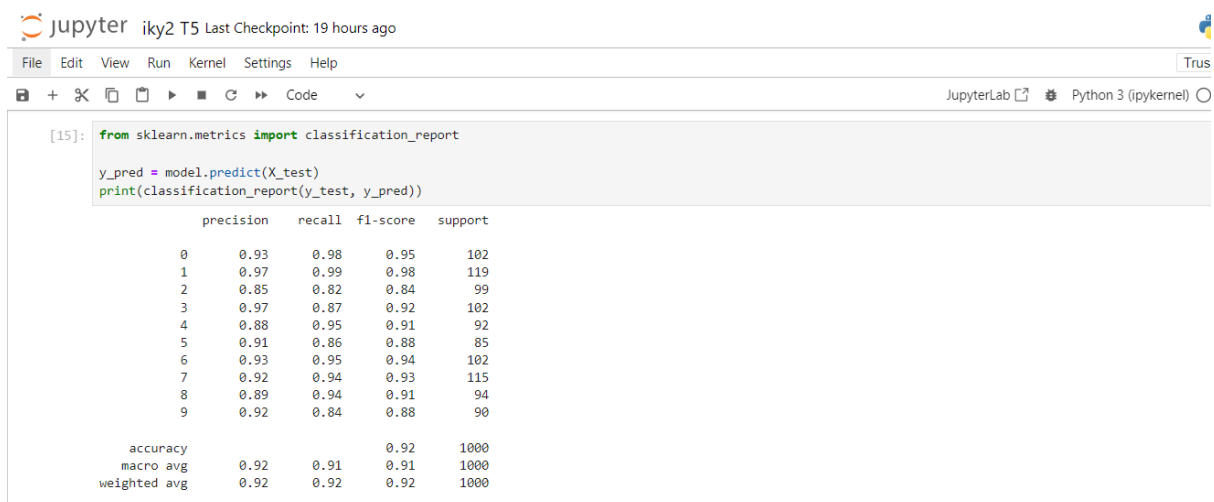
2.5. Klasifikasi dengan Support Vector Classifier | SVC



JupyterLab interface showing the training of an SVC model. The code cell [14] contains the following Python code:

```
[14]: from sklearn.svm import SVC  
  
model = SVC(random_state=0)  
model.fit(X_train, y_train)
```

The variable viewer shows the `SVC` object with `random_state=0`.



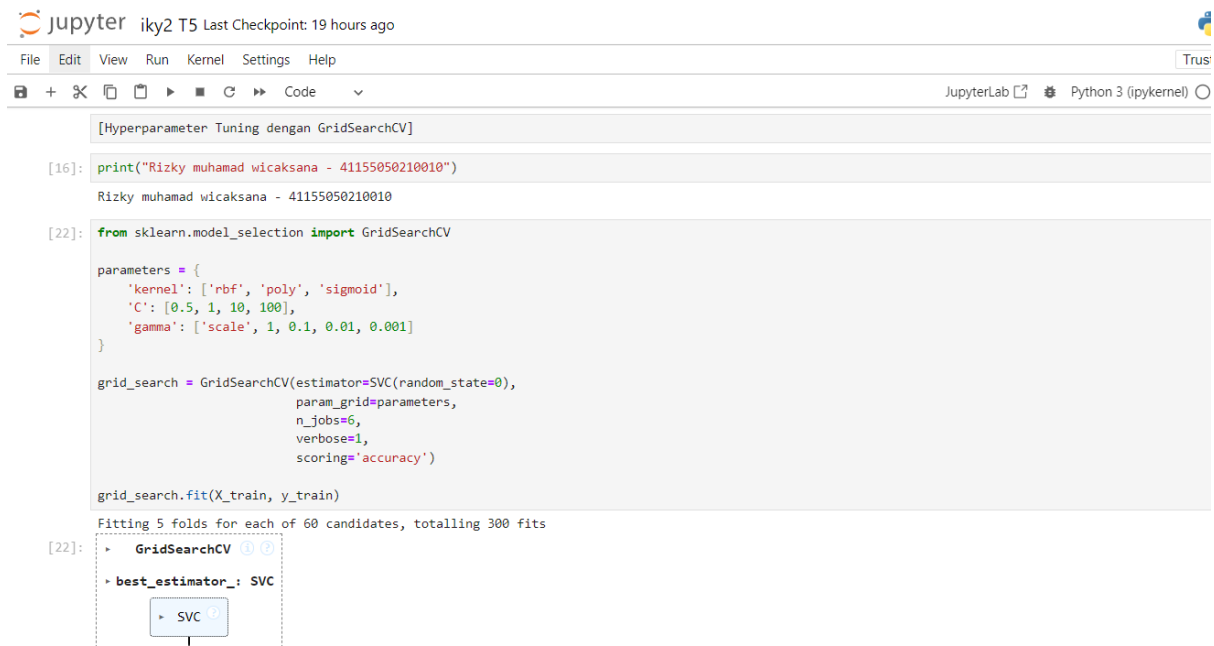
JupyterLab interface showing the classification report for the trained SVC model. The code cell [15] contains the following Python code:

```
[15]: from sklearn.metrics import classification_report  
  
y_pred = model.predict(X_test)  
print(classification_report(y_test, y_pred))
```

The output is a classification report showing precision, recall, f1-score, and support for each class (0-9) and overall metrics.

	precision	recall	f1-score	support
0	0.93	0.98	0.95	102
1	0.97	0.99	0.98	119
2	0.85	0.82	0.84	99
3	0.97	0.87	0.92	102
4	0.88	0.95	0.91	92
5	0.91	0.86	0.88	85
6	0.93	0.95	0.94	102
7	0.92	0.94	0.93	115
8	0.89	0.94	0.91	94
9	0.92	0.84	0.88	90
accuracy			0.92	1000
macro avg	0.92	0.91	0.91	1000
weighted avg	0.92	0.92	0.92	1000

2.6. Hyperparameter Tuning dengan Grid Search



JupyterLab interface showing the hyperparameter tuning of an SVC model using GridSearchCV. The code cell [22] contains the following Python code:

```
[22]: from sklearn.model_selection import GridSearchCV  
  
parameters = {  
    'kernel': ['rbf', 'poly', 'sigmoid'],  
    'C': [0.5, 1, 10, 100],  
    'gamma': ['scale', 1, 0.1, 0.01, 0.001]  
}  
  
grid_search = GridSearchCV(estimator=SVC(random_state=0),  
                           param_grid=parameters,  
                           n_jobs=6,  
                           verbose=1,  
                           scoring='accuracy')  
  
grid_search.fit(X_train, y_train)
```

The output shows the fitting process: "Fitting 5 folds for each of 60 candidates, totalling 300 fits".

The variable viewer shows the `GridSearchCV` object with `best_estimator_` set to `SVC`.

JupyterLab iky2 T5 Last Checkpoint: 19 hours ago

File Edit View Run Kernel Settings Help Trust

JupyterLab Python 3 (ipykernel)

```
[23]: print(f'Best Score: {grid_search.best_score_}')

best_params = grid_search.best_estimator_.get_params()
print(f'Best Parameters:')
for param in parameters:
    print(f'\t{param}: {best_params[param]}')

Best Score: 0.907
Best Parameters:
    kernel: rbf
    C: 10
    gamma: scale
```

2.7. Evaluasi Model

JupyterLab iky2 T5 Last Checkpoint: 19 hours ago

File Edit View Run Kernel Settings Help Trust

JupyterLab Python 3 (ipykernel)

[Predict & Evaluate]

```
[24]: print("Rizky muhamad wicaksana - 41155050210010")

Rizky muhamad wicaksana - 41155050210010

[25]: y_pred = grid_search.predict(X_test)

print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.93	0.98	0.96	102
1	0.98	0.99	0.98	119
2	0.87	0.85	0.86	99
3	0.99	0.89	0.94	102
4	0.91	0.95	0.93	92
5	0.92	0.89	0.90	85
6	0.93	0.94	0.94	102
7	0.93	0.93	0.93	115
8	0.89	0.95	0.92	94
9	0.92	0.88	0.90	90
accuracy			0.93	1000
macro avg	0.93	0.92	0.92	1000
weighted avg	0.93	0.93	0.93	1000