Optimasi Model Machine Learning dengan Python

Tujuan Pembelajaran:

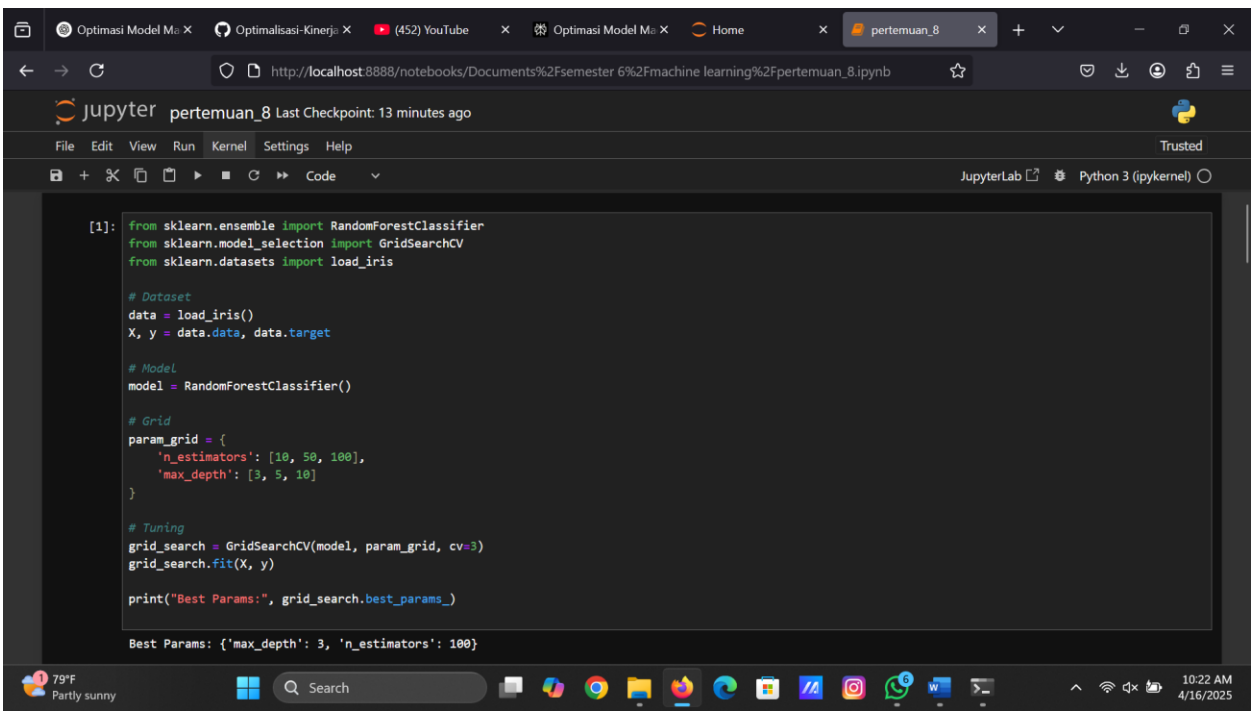
1. Memahami dan menerapkan hyperparameter tuning.
2. Melakukan feature engineering pada dataset.
3. Menerapkan ensemble methods.
4. Mengevaluasi kinerja model.
5. Menerapkan seluruh konsep dalam studi kasus optimasi model.

1. Pengenalan Hyperparameter Tuning
Hyperparameter adalah parameter yang ditentukan sebelum proses pelatihan dimulai. Contoh: n_estimators pada RandomForest.

Contoh Program: Hyperparameter Tuning dengan GridSearchCV

```
from sklearn.ensemble import RandomForestClassifier

from sklearn.model_selection import GridSearchCV

from sklearn.datasets import load_iris


# Dataset

data = load_iris()

X, y = data.data, data.target


# Model

model = RandomForestClassifier()


# Grid
```

```
param_grid = {
    'n_estimators': [10, 50, 100],
    'max_depth': [3, 5, 10]
}

# Tuning
grid_search = GridSearchCV(model, param_grid, cv=3)
grid_search.fit(X, y)

print("Best Params:", grid_search.best_params_)
```



2. Teknik-Teknik Feature Engineering
Feature engineering membantu model memahami data lebih baik dengan transformasi fitur atau menambah fitur baru.

Contoh Program: Scaling dan Encoding

```
import pandas as pd
```

```
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

# Data contoh
data = pd.DataFrame({
    'umur': [25, 30, 45],
    'jenis_kelamin': ['L', 'P', 'L']
})

# Tentukan kolom numerik dan kategorik
numeric_features = ['umur']
categorical_features = ['jenis_kelamin']

# Buat preprocessor (scaler + encoder)
numeric_transformer = Pipeline(steps=[
    ('scaler', StandardScaler())
])

categorical_transformer = Pipeline(steps=[
    ('onehot', OneHotEncoder(drop='first'))  # drop='first' agar tidak dummy trap
])

# Gabungkan semuanya dengan ColumnTransformer
```

```
preprocessor = ColumnTransformer(transformers=[
    ('num', numeric_transformer, numeric_features),
    ('cat', categorical_transformer, categorical_features)
])

# Transformasi fitur
X_transformed = preprocessor.fit_transform(data)

# Ambil nama kolom baru dari hasil transformasi
cat_feature_names = preprocessor.named_transformers_['cat'].named_steps['onehot'].get_feature_names_out(categorical_features)
all_feature_names = numeric_features + list(cat_feature_names)

# Buat DataFrame hasil transformasi
df_transformed = pd.DataFrame(X_transformed, columns=all_feature_names)

print("Data setelah feature engineering:")
print(df_transformed)
```

```
[3]: import pandas as pd
     from sklearn.preprocessing import StandardScaler, OneHotEncoder
     from sklearn.compose import ColumnTransformer
     from sklearn.pipeline import Pipeline

     # Data contoh
     data = pd.DataFrame({
         'umur': [25, 30, 45],
         'jenis_kelamin': ['L', 'P', 'L']
     })

     # Tentukan kolom numerik dan kategorik
     numeric_features = ['umur']
     categorical_features = ['jenis_kelamin']

     # Buat preprocessor (scaler + encoder)
     numeric_transformer = Pipeline(steps=[
         ('scaler', StandardScaler())
     ])

     categorical_transformer = Pipeline(steps=[
         ('onehot', OneHotEncoder(drop='first'))  # drop='first' agar tidak dummy trap
     ])

     # Gabungkan semuanya dengan ColumnTransformer
     preprocessor = ColumnTransformer(transformers=[
         ('num', numeric_transformer, numeric_features),
         ('cat', categorical_transformer, categorical_features)
     ])

     # Transformasi fitur
     X_transformed = preprocessor.fit_transform(data)

     # Ambil nama kolom baru dari hasil transformasi
     cat_feature_names = preprocessor.named_transformers_['cat'].named_steps['onehot'].get_feature_names_out(categorical_features)
     all_feature_names = numeric_features + list(cat_feature_names)

     # Buat DataFrame hasil transformasi
     df_transformed = pd.DataFrame(X_transformed, columns=all_feature_names)

     print("Data setelah feature engineering:")
     print(df_transformed)

     Data setelah feature engineering:
            umur  jenis_kelamin_P
     0 -0.980581              0.0
     1 -0.392232              1.0
     2  1.372813              0.0
```

3. Implementasi Ensemble Methods
Ensemble methods menggabungkan beberapa model agar hasil lebih akurat.

Contoh Program: RandomForest dan VotingClassifier

from sklearn.ensemble import RandomForestClassifier, VotingClassifier

from sklearn.linear_model import LogisticRegression

from sklearn.svm import SVC

from sklearn.datasets import load_iris

from sklearn.model_selection import train_test_split

Dataset

X, y = load_iris(return_X_y=True)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

Models

clf1 = LogisticRegression()

clf2 = RandomForestClassifier()

clf3 = SVC(probability=True)

Voting

ensemble = VotingClassifier(estimators=[

    ('lr', clf1), ('rf', clf2), ('svc', clf3)

], voting='soft')

ensemble.fit(X_train, y_train)

print("Accuracy:", ensemble.score(X_test, y_test))

```
[4]: from sklearn.ensemble import RandomForestClassifier, VotingClassifier
     from sklearn.linear_model import LogisticRegression
     from sklearn.svm import SVC
     from sklearn.datasets import load_iris
     from sklearn.model_selection import train_test_split

     # Dataset
     X, y = load_iris(return_X_y=True)
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

     # Models
     clf1 = LogisticRegression()
     clf2 = RandomForestClassifier()
     clf3 = SVC(probability=True)

     # Voting
     ensemble = VotingClassifier(estimators=[
         ('lr', clf1), ('rf', clf2), ('svc', clf3)
     ], voting='soft')

     ensemble.fit(X_train, y_train)
     print("Accuracy:", ensemble.score(X_test, y_test))

     Accuracy: 0.9555555555555556
```

4. Evaluasi Kinerja Model
Evaluasi model digunakan untuk mengetahui seberapa baik performa model.

Contoh Program: Confusion Matrix dan Classification Report

from sklearn.metrics import classification_report, confusion_matrix

y_pred = ensemble.predict(X_test)

print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))

print("Classification Report:\n", classification_report(y_test, y_pred))

```
]: from sklearn.metrics import classification_report, confusion_matrix

   y_pred = ensemble.predict(X_test)
   print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
   print("Classification Report:\n", classification_report(y_test, y_pred))

   Confusion Matrix:
    [[15  0  0]
    [ 0 18  2]
    [ 0  0 10]]
   Classification Report:
                 precision    recall  f1-score   support

              0       1.00      1.00      1.00        15
              1       1.00      0.90      0.95        20
              2       0.83      1.00      0.91        10

       accuracy                           0.96        45
      macro avg       0.94      0.97      0.95        45
   weighted avg       0.96      0.96      0.96        45
```

5. Studi Kasus Optimasi Model
Optimasi model klasifikasi untuk dataset breast cancer dari scikit-learn.

Contoh Program:

from sklearn.datasets import load_breast_cancer

from sklearn.ensemble import RandomForestClassifier

from sklearn.model_selection import GridSearchCV, train_test_split

from sklearn.metrics import classification_report

Load data

data = load_breast_cancer()

X_train, X_test, y_train, y_test = train_test_split(data.data, data.target, test_size=0.3)

Hyperparameter tuning

```
param_grid = {
    'n_estimators': [50, 100],
    'max_depth': [4, 6, 8]
}
grid = GridSearchCV(RandomForestClassifier(), param_grid, cv=3)
grid.fit(X_train, y_train)


# Evaluation
y_pred = grid.predict(X_test)
print("Best Params:", grid.best_params_)
print("Classification Report:\n", classification_report(y_test, y_pred))
```