Nama Kelompok : Tensorflow

Anggota :

- Mario Arya Mahardika / 220711664
- Octavionus Samuel Kusuma Wardana / 220711661
- Andreano Yong / 220711682
- Stewart Sidauruk / 220711816

Link Streamlit :

Klasifikasi : Random Forest

Code: https://projek-uts-pmdpmatensorflow-eqcahfappyzqpbw7tjyze8q.streamlit.app/

```
import pandas as pd
import numpy as np

#load data menggunakan functiion read_csv dari pandas
df_properti = pd.read_csv("D:\ATMA\sem 5\Mesin\Projek UTS Gasal 20242025-
20241016\Dataset UTS_Gasal 2425.csv")
df_properti.head(20)
```

```
#Mengecek data kosong, null, dan nan
print("data null\n", df_properti.isnull().sum())
print("\n")
print("data kosong \n", df_properti.empty)
print("\n")
print("data nan \n", df_properti.isna().sum())
```

```
df_properti=df_properti.drop('price', axis=1)
df_properti.head(50)
```

```
from sklearn.model_selection import train_test_split

X = df_properti.drop(columns=['category'], axis=1)
y = df_properti['category']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25,
random_state = 82)

print("bentuk X_train", X_train.shape)
print ("bentuk X_test", X_test.shape)
```

```python
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import make_column_transformer

cat_cols = ['hasyard', 'haspool', 'isnewbuilt',
            'hasstormprotector', 'hasstorageroom']

col_transformer = make_column_transformer(
    (OneHotEncoder(), cat_cols),
    remainder='passthrough'
)

X_train_enc = col_transformer.fit_transform(X_train)
X_test_enc = col_transformer.transform(X_test)

df_train_enc = pd.DataFrame(X_train_enc,
columns=col_transformer.get_feature_names_out())
df_test_enc = pd.DataFrame(X_test_enc,
columns=col_transformer.get_feature_names_out())

print("Kolom kategorik:", cat_cols)
print(df_train_enc.columns)
print(df_train_enc.head(10))
print(df_test_enc.head(10))
```

```python
#buat varibel baru untk menampung hadil transformasi kolom
X_train_enc=col_transformer.fit_transform(X_train)
#khusus untuk train set gunakan fungsi fit_transform, untuk test set gunakan
transform saja
X_test_enc=col_transformer.fit_transform(X_test)

#jika ingin melihat hasil dari transformasi, muat dalam dataframe
df_train_enc=pd. DataFrame(X_train_enc,
columns=col_transformer.get_feature_names_out())
df_test_enc=pd. DataFrame(X_test_enc,
columns=col_transformer.get_feature_names_out())

df_train_enc.head(10)
df_test_enc.head(10)
```

```python
# import Library yang dibutuhkan
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.feature_selection import SelectKBest, SelectPercentile
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV, StratifiedKFold
import numpy as np

# buat rancangan pipeline mulai dari data scaling hingga classifier
pipe_RF = [('data scaling', StandardScaler()),
           ('feature select', SelectKBest()),
           ('clf', RandomForestClassifier(random_state=0,
class_weight='balanced'))]  # random_state pakai 2 digit npm terakhir

# buat parameter grid untuk step feature selection dan classifier
params_grid_RF = [
    {
        'data scaling': [StandardScaler()],
        'feature select__k': np.arange(2, 6),
        'clf__max_depth': np.arange(4, 5),
        'clf__n_estimators': [100, 150]
    },
    {
        'data scaling': [StandardScaler()],
        'feature select': [SelectPercentile()],
        'feature select__percentile': np.arange(20, 50),
        'clf__max_depth': np.arange(4, 5),
        'clf__n_estimators': [100, 150]
    },
    {
        'data scaling': [MinMaxScaler()],
        'feature select__k': np.arange(2, 6),
        'clf__max_depth': np.arange(4, 5),
        'clf__n_estimators': [100, 150]
    },
    {
        'data scaling': [MinMaxScaler()],
        'feature select': [SelectPercentile()],
        'feature select__percentile': np.arange(20, 50),
        'clf__max_depth': np.arange(4, 5),
        'clf__n_estimators': [100, 150]
    }
]

# Buat pipeline
estimator_RF = Pipeline(pipe_RF)

# Muat pipeline dan parameter grid ke dalam objek GridSearchCV dengan
Stratified 5-fold CV
```

```
SKF = StratifiedKFold(n_splits=5, shuffle=True, random_state=82)

GSCV_RF = GridSearchCV(estimator_RF, params_grid_RF, cv=SKF)

# Jalankan objek GridSearchCV untuk melatih model dengan train set menggunakan
fungsi fit
GSCV_RF.fit(X_train_enc, y_train)
print("GSCV training finished")
```

```
# Import libraries yang diperlukan
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay,
classification_report
import matplotlib.pyplot as plt

# Tampilkan skor cross-validation
print("CV Score: {}".format(GSCV_RF.best_score_))

# Tampilkan skor model terbaik GSCV pada test set
print("Test Score: {}".format(GSCV_RF.best_estimator_.score(X_test_enc,
y_test)))

# Tampilkan best model dan best features
print("Best model:", GSCV_RF.best_estimator_)

# Mask untuk menampilkan fitur terbaik, jika 'SelectKBest' atau
'SelectPercentile' digunakan
mask = GSCV_RF.best_estimator_.named_steps['feature select'].get_support()
print("Best features:", df_train_enc.columns[mask])

# Buat prediksi dari test set
RF_pred = GSCV_RF.predict(X_test_enc)

# Buat confusion matrix
cm = confusion_matrix(y_test, RF_pred, labels=GSCV_RF.classes_)

# Buat confusion matrix display
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=GSCV_RF.classes_)
disp.plot()

# Berikan judul pada confusion matrix plot
plt.title("Random Forest Confusion Matrix")
plt.show()

# Tampilkan classification report
print("Classification report RF: \n", classification_report(y_test, RF_pred))
```

Klasifikasi : Logistic Regression

```
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.feature_selection import SelectKBest, SelectPercentile
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV, StratifiedKFold
import numpy as np

# Buat pipeline mulai dari data scaling hingga classifier
pipe_LR = [
    ('data scaling', StandardScaler()),
    ('feature select', SelectKBest()),
    ('clf', LogisticRegression(random_state=0, max_iter=1000,
class_weight='balanced'))
]

# Buat parameter grid untuk step feature selection dan classifier
params_grid_LR = [
    {
        'data scaling': [StandardScaler()],
        'feature select__k': np.arange(2, 6),
        'clf__C': [0.01, 0.1, 1, 10],  # Regularization parameter C
        'clf__penalty': ['l1', 'l2'],   # Regularization type
        'clf__solver': ['liblinear', 'saga']  # Solver untuk l1 dan l2
    },
    {
        'data scaling': [StandardScaler()],
        'feature select': [SelectPercentile()],
        'feature select__percentile': np.arange(20, 50),
        'clf__C': [0.01, 0.1, 1, 10],
        'clf__penalty': ['l1', 'l2'],
        'clf__solver': ['liblinear', 'saga']
    },
    {
        'data scaling': [MinMaxScaler()],
        'feature select__k': np.arange(2, 6),
        'clf__C': [0.01, 0.1, 1, 10],
        'clf__penalty': ['l1', 'l2'],
        'clf__solver': ['liblinear', 'saga']
    },
    {
        'data scaling': [MinMaxScaler()],
        'feature select': [SelectPercentile()],
        'feature select__percentile': np.arange(20, 50),
```

```python
        'clf__C': [0.01, 0.1, 1, 10],
        'clf__penalty': ['l1', 'l2'],
        'clf__solver': ['liblinear', 'saga']
    }
]

# Buat pipeline
estimator_LR = Pipeline(pipe_LR)

# Muat pipeline dan parameter grid ke dalam objek GridSearchCV dengan
Stratified 5-fold CV
SKF = StratifiedKFold(n_splits=5, shuffle=True, random_state=82)

GSCV_LR = GridSearchCV(estimator_LR, params_grid_LR, cv=SKF)

# Jalankan objek GSCV untuk melatih model dengan train set menggunakan fungsi
fit
GSCV_LR.fit(X_train_enc, y_train)

# Tampilkan hasil evaluasi
print("CV Score: {}".format(GSCV_LR.best_score_))
print("Test Score: {}".format(GSCV_LR.best_estimator_.score(X_test_enc,
y_test)))
print("Best model:", GSCV_LR.best_estimator_)
```

```python
# Import libraries yang diperlukan
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay,
classification_report
import matplotlib.pyplot as plt

# Tampilkan skor cross-validation
print("CV Score: {}".format(GSCV_LR.best_score_))

# Tampilkan skor model terbaik GSCV pada test set
print("Test Score: {}".format(GSCV_LR.best_estimator_.score(X_test_enc,
y_test)))

# Tampilkan best model dan best features
print("Best model:", GSCV_LR.best_estimator_)

# Mask untuk menampilkan fitur terbaik, jika 'SelectKBest' atau
'SelectPercentile' digunakan
mask = GSCV_LR.best_estimator_.named_steps['feature select'].get_support()
print("Best features:", df_train_enc.columns[mask])
```

```
# Buat prediksi dari test set
LR_pred = GSCV_LR.predict(X_test_enc)

# Buat confusion matrix
cm = confusion_matrix(y_test, LR_pred, labels=GSCV_LR.classes_)

# Buat confusion matrix display
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=GSCV_LR.classes_)
disp.plot()

# Berikan judul pada confusion matrix plot
plt.title("Logistic Regression Confusion Matrix")
plt.show()

# Tampilkan classification report
print("Classification report LR: \n", classification_report(y_test, LR_pred))
```

Klasifikasi SVM :

Code:

```
import pandas as pd
import numpy as np

#load data menggunakan functiion read_csv dari pandas
df_properti=pd.read_csv(r'C:\UTS_Bagian_SVM_GBT\Dataset UTS_Gasal 2425.csv')
df_properti.head(20)
```

```
#Mengecek data kosong, null, dan nan

print("data null\n", df_properti.isnull().sum())
print("\n")
print("data kosong \n", df_properti.empty)
print("\n")
print("data nan \n", df_properti.isna().sum())
```

```
df_properti=df_properti.drop('price', axis=1)
df_properti.head(50)
```

```
from sklearn.model_selection import train_test_split

# Misalkan df_properti adalah DataFrame yang berisi data yang sudah Anda load
```

```
x = df_properti.drop(columns=['category'], axis=1)  # ganti 'target_column'
dengan nama kolom target Anda
y = df_properti['category']  # kolom target

# Pisahkan data menjadi train dan test
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25,
random_state=82)

print("bentuk X_train", x_train.shape)
print ("bentuk X_test", x_test.shape)
```

```
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import make_column_transformer

cat_cols = ['hasyard', 'haspool', 'isnewbuilt',
            'hasstormprotector', 'hasstorageroom']

col_transformer = make_column_transformer(
    (OneHotEncoder(), cat_cols),
    remainder='passthrough'
)

x_train_enc = col_transformer.fit_transform(x_train)
x_test_enc = col_transformer.transform(x_test)

df_train_enc = pd.DataFrame(x_train_enc,
columns=col_transformer.get_feature_names_out())
df_test_enc = pd.DataFrame(x_test_enc,
columns=col_transformer.get_feature_names_out())

print("Kolom kategorik:", cat_cols)
print(df_train_enc.columns)
print(df_train_enc.head(10))
print(df_test_enc.head(10))
```

```
#buat varibel baru untk menampung hadil transformasi kolom
x_train_enc=col_transformer.fit_transform(x_train)
#khusus untuk train set gunakan fungsi fit_transform, untuk test set gunakan
transform saja
x_test_enc=col_transformer.fit_transform(x_test)

#jika ingin melihat hasil dari transformasi, muat dalam dataframe
df_train_enc=pd. DataFrame(x_train_enc,
columns=col_transformer.get_feature_names_out())
df_test_enc=pd. DataFrame(x_test_enc,
columns=col_transformer.get_feature_names_out())

df_train_enc.head(10)
df_test_enc.head(10)
```

```
#import Library yang dibutuhkan untuk pipeline, GSCV, dan metrik evaluasi
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.feature_selection import SelectPercentile ,SelectKBest
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV, StratifiedKFold
from sklearn.pipeline import Pipeline
from sklearn.metrics import classification_report, confusion_matrix,
ConfusionMatrixDisplay

#buat rancangan pipeline mulai dari data scaling hingga classifier
pipe_svm = Pipeline(steps=[
    ('scale', MinMaxScaler()),
    ('feat_select', SelectKBest()),
    ('clf', SVC(class_weight='balanced'))
])

#buat parameter grid untuk step feature selection dan classifier
params_grid_svm = [
    {
    'scale': [MinMaxScaler()],
    'feat_select__k':np.arange(2,6),
    'clf__kernel': ['poly','rbf'],
    'clf__C':[0.1,1],
    'clf__gamma':[0.1, 1]


    },
    {
        'scale': [MinMaxScaler()],
        'feat_select': [SelectPercentile()],
        'feat_select__percentile':np.arange(20,50),
        'clf__kernel': ['poly','rbf'],
         'clf__C':[ 0.1, 1],
    'clf__gamma':[0.1, 1]
    },
    {
    'scale': [StandardScaler()],
    'feat_select__k':np.arange(2,6),
    'clf__kernel': ['poly', 'rbf'],
     'clf__C':[0.1, 1],
    'clf__gamma':[0.1, 1]
    },
    {
        'scale': [StandardScaler()],
        'feat_select': [SelectPercentile()],
        'feat_select__percentile':np.arange(20,50),
        'clf__kernel': ['poly','rbf'],
         'clf__C':[0.1, 1],
    'clf__gamma':[0.1, 1]
    }
]

estimator_svm = Pipeline(pipe_svm)
SKF = StratifiedKFold(n_splits=5, shuffle=True, random_state=82)

GSCV_SVM = GridSearchCV(pipe_svm, params_grid_svm, cv=SKF)

GSCV_SVM.fit(x_train_enc, y_train)
print("GSCV training finished")
```

```
#tampilkan skor cross-validation
print("CV Score : {}".format(GSCV_SVM.best_score_))
#tampilkan skor model terbaik GSCV pada test set
print("Test Score: {}".format(GSCV_SVM.best_estimator_.score(x_test_enc,
y_test)))
#tampilkan best model dan best features
print("Best model:", GSCV_SVM.best_estimator_)
mask = GSCV_SVM.best_estimator_.named_steps['feat_select'].get_support()
print("Best features:", df_train_enc.columns[mask])

#buat prediksi dari test set
SVM_pred = GSCV_SVM.predict(x_test_enc)

import matplotlib.pyplot as plt
#buat confusion matrix
cm = confusion_matrix(y_test, SVM_pred, labels=GSCV_SVM.classes_)
#buat confusion matrix display
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=GSCV_SVM.classes_)
disp.plot()
plt.title("SVM Confusion Matrix")
plt.show()

#tampilkan classification report
print("Classification report SVM:\n", classification_report(y_test, SVM_pred))
```

Klasifikasi : GBT

Code:

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.feature_selection import SelectFromModel
from sklearn.tree import DecisionTreeClassifier

pipe_GBT = Pipeline(steps=[
    ('feat_select', SelectKBest()),
    ('clf', GradientBoostingClassifier(random_state=82))])

params_grid_GBT = [
    {
        'feat_select__k': np.arange(2,6),
    'clf__max_depth': [*np.arange(4,5)],
    'clf__n_estimators': [100,150],
    'clf__learning_rate': [0.01,0.1,1]
    },
    {
        'feat_select': [SelectPercentile()],
        'feat_select__percentile':np.arange(20,50),
        'clf__max_depth': [*np.arange(4,5)],
        'clf__n_estimators': [100,150],
        'clf__learning_rate': [0.01,0.1,1]
    },
    {
        'feat_select__k': np.arange(2,6),
    'clf__max_depth': [*np.arange(4,5)],
    'clf__n_estimators': [100,150],
    'clf__learning_rate': [0.01,0.1,1]
    },
    {
        'feat_select': [SelectPercentile()],
        'feat_select__percentile':np.arange(20,50),
        'clf__max_depth': [*np.arange(4,5)],
        'clf__n_estimators': [100,150],
        'clf__learning_rate' : [0.01,0.1,1]
    }
]

GSCV_GBT = GridSearchCV(pipe_GBT, params_grid_GBT,
cv=StratifiedKFold(n_splits=5))
GSCV_GBT.fit(x_train_enc, y_train)
print("GSCV Finished")
```

```
#tampilkan skor cross-validation
print("CV Score: {}".format(GSCV_GBT.best_score_))
#tampilkan skor model terbaik GSCV pada test set
print("Test Score: {}".format(GSCV_GBT.best_estimator_.score(x_test_enc,
y_test)))
#tampilkan best model dan best features
print("Best model:", GSCV_GBT.best_estimator_)

mask = GSCV_GBT.best_estimator_.named_steps['feat_select' ].get_support()
print("Best features:", df_train_enc.columns[mask])

#buat prediksi dari test set
RF_pred = GSCV_GBT.predict(x_test_enc)

import matplotlib.pyplot as plt
#buat confusion matrix
cm = confusion_matrix(y_test, RF_pred, labels=GSCV_GBT.classes_)
#buat confusion matrix display
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=GSCV_GBT.classes_)
disp.plot()

plt.title("GBT Confusion Matrix")
plt.show()
#tampilkan Classification report
print("Classification report GBT: \n", classification_report(y_test, RF_pred))
```

2.Regresi:

Regresi: Ridge Regression

Code:

```
import pandas as pd
import numpy as np

df_price=pd.read_csv('D:\Download 4\ML_UTS\Dataset UTS_Gasal 2425.csv')
df_price.head(20)
```

```
df_price2=df_price.drop('category', axis=1)
df_price2.head(20)
```

```
df_price2['price'].value_counts()
```

```
print("data null \n",df_price2.isnull().sum())
print("\ndata kosong \n",df_price2.empty)
print("\ndata nan \n",df_price2.isna().sum())
```

```
print("Sebelum Pengecekan data duplikat, ", df_price2.shape)
df_price3=df_price2.drop_duplicates(keep='last')
print("Setelah Pengecekan data duplikat, ", df_price3.shape)
```

```
from sklearn.model_selection import train_test_split

x_regress = df_price3.drop('price', axis=1)
y_regress = df_price3['price']
x_train_price, x_test_price, y_train_price, y_test_price =
train_test_split(x_regress, y_regress, test_size=0.25, random_state=82)

print(x_train_price.shape)
print(x_test_price.shape)
```

```
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import make_column_transformer

kolom_kategori=['hasyard','haspool','isnewbuilt','hasstormprotector','hasstora
geroom']

transform = make_column_transformer(
    (OneHotEncoder(),kolom_kategori),remainder='passthrough'
)
```

```
x_train_enc=transform.fit_transform(x_train_price)
x_test_enc=transform.fit_transform(x_test_price)

df_train_enc=pd.DataFrame(x_train_enc,columns=transform.get_feature_names_out(
))
df_test_enc=pd.DataFrame(x_test_enc,columns=transform.get_feature_names_out())

df_train_enc.head(10)
df_test_enc.head(10)
```

```
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import SelectPercentile, f_regression
from sklearn.metrics import mean_absolute_error, mean_squared_error
import numpy as np

```

```python
pipe_Ridge = Pipeline(steps=[
            ('scale', StandardScaler()),
            ('feature_selection', SelectPercentile(score_func=f_regression)),
            ('reg', Ridge())
            ])

param_grid_Ridge = {
    'reg__alpha': [0.01, 0.1, 1, 10, 100],
    'feature_selection__percentile': np.arange(1, 101)
}

GSCV_RR = GridSearchCV(pipe_Ridge, param_grid_Ridge, cv=5,
                            scoring='neg_mean_squared_error',
error_score='raise')

GSCV_RR.fit(x_train_enc, y_train_price)

print("Best model: {}".format(GSCV_RR.best_estimator_))
print("Ridge best parameters: {}".format(GSCV_RR.best_params_))
print("Koefisien/bobot:
{}".format(GSCV_RR.best_estimator_.named_steps['reg'].coef_))
print("Intercept/bias:
{}".format(GSCV_RR.best_estimator_.named_steps['reg'].intercept_))

Ridge_predict = GSCV_RR.predict(x_test_enc)

mse_Ridge = mean_squared_error(y_test_price, Ridge_predict)
mae_Ridge = mean_absolute_error(y_test_price, Ridge_predict)

print("Ridge Mean Squared Error (MSE): {}".format(mse_Ridge))
print("Ridge Mean Absolute Error (MAE): {}".format(mae_Ridge))
print("Ridge Root Mean Squared Error: {}".format(np.sqrt(mse_Ridge)))
```

```python
df_results = pd.DataFrame(y_test_price, columns=['price'])
df_results = pd.DataFrame(y_test_price)
df_results['Ridge Prediction'] = Ridge_predict

df_results['Selisih_price_RR'] = df_results['Ridge Prediction'] -
df_results['price']

df_results.head()
```

```python
df_results.describe()
```

Regresi: SVR

```python
from sklearn.svm import SVR
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import SelectKBest, f_regression
from sklearn.metrics import mean_absolute_error, mean_squared_error

pipe_SVR = Pipeline(steps=[
            ('scale', StandardScaler()),
            ('feature_selection', SelectKBest(score_func=f_regression)),
            ('reg', SVR(kernel='linear'))
            ])

param_grid_SVR = {
    'reg__C': [0.01,0.1,1,10,100],
    'reg__epsilon': [0.1,0.2,0.5,1],
    'feature_selection__k': np.arange(1,20)
}

GSCV_SVR = GridSearchCV(pipe_SVR, param_grid_SVR, cv=5,
scoring='neg_mean_squared_error')

GSCV_SVR.fit(x_train_enc, y_train_price)

print("Best model:{}".format(GSCV_SVR.best_estimator_))
print("SVR best parameters:{}".format(GSCV_SVR.best_params_))
print("Koefisien/bobot:{}".format(GSCV_SVR.best_estimator_.named_steps['reg'].
coef_))
print("Intercept/bias:{}".format(GSCV_SVR.best_estimator_.named_steps['reg'].i
ntercept_))

SVR_predict = GSCV_SVR.predict(x_test_enc)

mse_SVR = mean_squared_error(y_test_price, SVR_predict)
mae_SVR = mean_absolute_error(y_test_price, SVR_predict)

print("SVR Maan Squared Error (MSE): {}".format(mse_SVR))
print("SVR Mean Absolute Error (MAE): {}".format(mae_SVR))
print("SVR Root Mean Squared Error: {}".format(np.sqrt(mse_SVR)))
```

```python
df_results['SVR Prediction'] = SVR_predict
df_results = pd.DataFrame(y_test_price)
df_results['SVR Prediction'] = SVR_predict
df_results['Selisih_price_SVR'] = df_results['SVR Prediction'] -
df_results['price']
```

```
df_results.head()
```

```
df_results.describe()
```

```
df_results = pd.DataFrame({'price': y_test_price})

df_results['Ridge Prediction'] = Ridge_predict
df_results['Selisih_price_RR'] = df_results['price'] - df_results['Ridge
Prediction']

df_results['SVR Prediction'] = SVR_predict
df_results['Selisih_price_SVR'] = df_results['price'] - df_results['SVR
Prediction']

df_results.head()
from sklearn.metrics import mean_absolute_error, mean_squared_error
import numpy as np

mae_ridge = mean_absolute_error(df_results['price'], df_results['Ridge
Prediction'])
rmse_ridge = np.sqrt(mean_squared_error(df_results['price'], df_results['Ridge
Prediction']))
ridge_feature_count = GSCV_RR.best_params_['feature_selection__percentile']

mae_svr = mean_absolute_error(df_results['price'], df_results['SVR
Prediction'])
rmse_svr = np.sqrt(mean_squared_error(df_results['price' ], df_results['SVR
Prediction']))
svr_feature_count = GSCV_SVR.best_params_['feature_selection__k']



print(f"Ridge MAE: {mae_ridge}, Ridge RMSE: {rmse_ridge}, Ridge Feature Count:
{ridge_feature_count}")
print(f"SVR MAE: {mae_svr}, SVR RMSE: {rmse_svr}, SVR Feature Count:
{svr_feature_count}")
```

```
import matplotlib.pyplot as plt

plt.figure(figsize=(20,5))
data_len = range(len(y_test_price))
plt.scatter(data_len, df_results. price, label="actual", color="blue")
plt.plot(data_len, df_results['Ridge Prediction'], label="Ridge Prediction",
color="green", linewidth=4, linestyle="dashed")
plt.plot(data_len, df_results['SVR Prediction'], label="SVR Prediction",
color="yellow", linewidth=2, linestyle="-.")
```

```
plt.legend()
plt.show
```

```
import pickle

best_model = GSCV_RR.best_estimator_

with open('Ridge_price_model.pkl', 'wb') as f:
    pickle.dump(best_model, f)

print("Model terbaik berhasil disimpan ke 'Ridge_price_model.pkl'")
```

Regresi: Lasso regression

Code:

```
from sklearn.linear_model import Lasso
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import SelectPercentile, f_regression
from sklearn.metrics import mean_absolute_error, mean_squared_error

pipe_Lasso = Pipeline(steps=[
            ('scale', StandardScaler()),
            ('feature_selection', SelectPercentile(score_func=f_regression)),
            ('reg', Lasso(max_iter=1000))
            ])

param_grid_Lasso = {
    'reg__alpha': [0.01,0.1,1,10,100],
    'feature_selection__percentile': np.arange(10,100,10)
}

GSCV_Lasso = GridSearchCV(pipe_Lasso, param_grid_Lasso, cv=5,
scoring='neg_mean_squared_error')

# GSCV_Lasso.fit(X_train_price, y_train_price)
GSCV_Lasso.fit(X_train_enc, y_train_price)
#GSCV_RR.fit(X_train_enc, y_train_price)

print("Best model:{}".format(GSCV_Lasso.best_estimator_))
print("Lasso best paramaters:{}".format(GSCV_Lasso.best_params_))

print("Koefisien/bobot:{}".format(GSCV_Lasso.best_estimator_.named_steps['reg'
].coef_))
```

```
print("Intecept/bias:{}".format(GSCV_Lasso.best_estimator_.named_steps['reg'].
intercept_))

Lasso_predict = GSCV_Lasso.predict(X_test_enc)

mse_Lasso = mean_squared_error(y_test_price, Lasso_predict)
mae_Lasso = mean_absolute_error(y_test_price, Lasso_predict)

print("Lasso Mean Squared Error (MSE): {}".format(mse_Lasso))
print("Lasso Mean Absolute Error (MAE): {}".format(mae_Lasso))
print("Lasso Root Mean Squared Error: {}".format(np.sqrt(mse_Lasso)))
```

Regresi: Random Forest Regressor

Code:

```
 from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import SelectKBest, f_regression
from sklearn.metrics import mean_absolute_error, mean_squared_error
import numpy as np


pipe_RFR = Pipeline(steps=[
            ('scale', StandardScaler()),
            ('feature_selection', SelectKBest(score_func=f_regression)),
            ('reg', RandomForestRegressor())
            ])


param_grid_RFR = {
    'reg__n_estimators': [100, 200, 300],
    'reg__max_depth': [3, 4, 5],
    'reg__min_samples_split': [2, 5, 10],
    'feature_selection__k': np.arange(1, 20)
}


GSCV_RFR = GridSearchCV(pipe_RFR, param_grid_RFR, cv=5,
scoring='neg_mean_squared_error')


GSCV_RFR.fit(X_train_enc, y_train_price)

# Best estimator and parameters
```

```
print("Best model:{}".format(GSCV_RFR.best_estimator_))
print("RFR best paramaters:{}".format(GSCV_RFR.best_params_))

# Feature importances
print("Feature Importances:
{}".format(GSCV_RFR.best_estimator_.named_steps['reg'].feature_importances_))

# Predictions
RFR_predict = GSCV_RFR.predict(X_test_enc)


mse_RFR = mean_squared_error(y_test_price, RFR_predict)
mae_RFR = mean_absolute_error(y_test_price, RFR_predict)

print("RFR Mean Squared Error (MSE): {}".format(mse_RFR))
print("RFR Mean Absolute Error (MAE): {}".format(mae_RFR))
print("RFR Root Mean Squared Error: {}".format(np.sqrt(mse_RFR)))
```