Regression Task

- Regression with Decision Tree

```
# Bagi data menjadi set pelatihan dan set pengujian
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Pilih algoritma Decision Tree Regressor
model = DecisionTreeRegressor()

# Latih model
model.fit(X_train, y_train)

# Prediksi menggunakan data pengujian
y_pred = model.predict(X_test)

# Evaluasi model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse}')
print(f'R-squared: {r2}')
```

Output:

```
Mean Squared Error      : 0.37918885606667274
R-squared               : 0.9999479735693169
```

```
from sklearn.tree import export_graphviz
import graphviz

# Export the decision tree to a dot file
dot_data = export_graphviz(model, out_file=None,
                           feature_names=X.columns,
                           filled=True, rounded=True,
special_characters=True)

# Visualize the decision tree using graphviz
graph = graphviz.Source(dot_data)
graph.render("DecisionTreeVisualization")  # Save the visualization to a
file
```

```
graph.view("DecisionTreeVisualization")  # Open the visualization in a new
tab
```

Output:

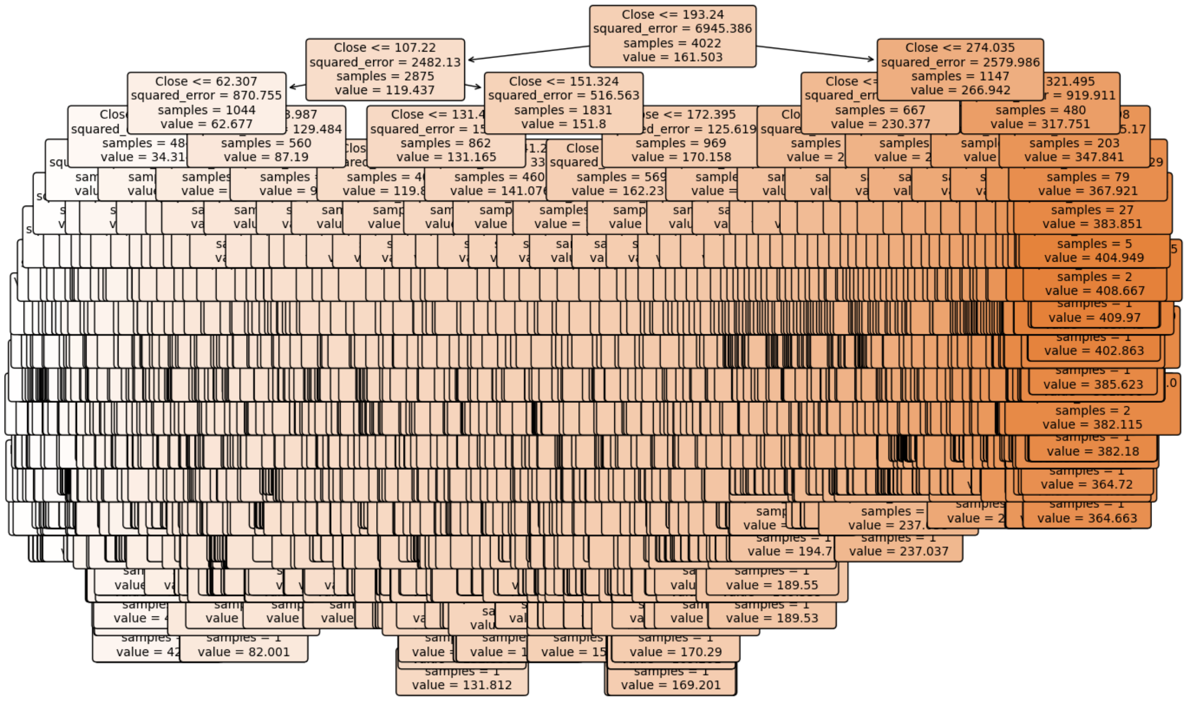
'DecisionTreeVisualization.pdf'

```
from sklearn.tree import DecisionTreeRegressor, plot_tree
import matplotlib.pyplot as plt

# Assuming you have already trained your model as per your code
# Decision Tree Regressor
model = DecisionTreeRegressor()

# Train the model
model.fit(X_train, y_train)

# Plot the decision tree
plt.figure(figsize=(15, 10))
plot_tree(model, filled=True, feature_names=X.columns, rounded=True,
fontsize=10)
plt.show()
```



File decision tree terpisah (nama file = 'DecisionTreeVisualization.pdf')

- Regression with KNN

```
# Lakukan scaling pada fitur
scaler = preprocessing.StandardScaler()
X_scaled = scaler.fit_transform(X)

# Bagi data menjadi set pelatihan dan set pengujian
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y,
test_size=0.2, random_state=42)

# Pilih algoritma K-Nearest Neighbors Regressor
k_value = 5  # Ganti dengan nilai K yang diinginkan
model = KNeighborsRegressor(n_neighbors=k_value)

# Latih model
model.fit(X_train, y_train)

# Prediksi menggunakan data pengujian
y_pred = model.predict(X_test)

# Evaluasi model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse}')
print(f'R-squared: {r2}')
```

Output:

```
Mean Squared Error     : 31.053516916832347
R-squared              : 0.9957393166505478
```

- Regression with Linear Regression

```
# Lakukan scaling pada fitur
scaler = preprocessing.StandardScaler()
X_scaled = scaler.fit_transform(X)

# Bagi data menjadi set pelatihan dan set pengujian
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y,
test_size=0.2, random_state=42)

# Pilih algoritma Linear Regression
model = LinearRegression()
```

```python
# Latih model
model.fit(X_train, y_train)

# Prediksi menggunakan data pengujian
y_pred = model.predict(X_test)

# Evaluasi model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse}')
print(f'R-squared: {r2}')
```

Output:

```
Mean Squared Error     : 0.4318781972782903
R-squared              : 0.9999407443527552
```