20250420_01

April 20, 2025

[62]:
```python
# Creating the DataFrame
import pandas as pd

data = {'student_id': ['S01', 'S02', 'S03', 'S04', 'S05', 'S06', 'S07', 'S08',␣
 ↪'S09', 'S10'],
        'math_score': [80, 92, 70, 88, 60, 95, 75, 85, 66, 78],
        'english_score': [78, 85, 75, 90, 65, 93, 68, 88, 70, 80],
        'gender': ['F', 'M', 'F', 'M', 'F', 'M', 'F', 'M', 'F', 'M'],
        'school_type': ['Public', 'Private', 'Public', 'Private', 'Private',
                        'Public', 'Private', 'Public', 'Public', 'Private'],
        'final_score': [82, 94, 73, 92, 62, 97, 76, 90, 68, 81]}

df = pd.DataFrame(data)
```

[64]:
```python
# Feature engineering
num_cols = ['math_score', 'english_score']
cat_cols = ['gender', 'school_type']

# Checking if any potential problems
{col: df[col].unique() for col in cat_cols}
```

[64]:
```
{'gender': array(['F', 'M'], dtype=object),
 'school_type': array(['Public', 'Private'], dtype=object)}
```

[66]:
```python
# Preprocessing
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer

preprocessor = ColumnTransformer([('num', StandardScaler(), num_cols),
                                  ('cat', OneHotEncoder(drop = 'first'),␣
 ↪cat_cols)])
```

[68]:
```python
# Cleaning the data and making training/test sets
from sklearn.model_selection import train_test_split

X = df.drop(columns = ['student_id', 'final_score'])
y = df['final_score']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3,␣
↪random_state = 42)
```

[70]:
```
# Creating pipelines : Linear Regression and Decision Tree
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor

# Linear Regression
pipe_lr = Pipeline([('preprocessing', preprocessor),
                    ('model', LinearRegression())])

# Decision Tree
pipe_tree = Pipeline([('preprocessing', preprocessor),
                      ('model', DecisionTreeRegressor(random_state=42))])
```

[72]:
```
# Trainging
pipe_lr.fit(X_train, y_train)
pipe_tree.fit(X_train, y_train)

# Predicting
y_pred_lr = pipe_lr.predict(X_test)
y_pred_tree = pipe_tree.predict(X_test)
```

[74]:
```
# Evaluating
from sklearn.metrics import mean_absolute_error, root_mean_squared_error,␣
↪r2_score

# Linear Regression
mae_lr = mean_absolute_error(y_test, y_pred_lr)
rmse_lr = root_mean_squared_error(y_test, y_pred_lr)
r2_lr = r2_score(y_test, y_pred_lr)

# Decision Tree
mae_tree = mean_absolute_error(y_test, y_pred_tree)
rmse_tree = root_mean_squared_error(y_test, y_pred_tree)
r2_tree = r2_score(y_test, y_pred_tree)

# Printing the results
print("Linear Regression")
print("MAE:", mae_lr)
print("RMSE:", rmse_lr)
print("R²:", r2_lr)

print("\nDecision Tree")
print("MAE:", mae_tree)
print("RMSE:", rmse_tree)
```

```
print("R²:", r2_tree)
```

```
Linear Regression
MAE: 1.2464561969668135
RMSE: 1.4534331865384456
R²: 0.987541145314822

Decision Tree
MAE: 4.666666666666667
RMSE: 5.0990195135927845
R²: 0.8466579292267365
```

In the baseline model using original features (math_score, english_score, gender, school_type), the Linear Regression model outperformed the Decision Tree with an MAE of 1.25 vs 4.67 and an R^2 of 0.99 vs 0.85. This suggests that the relationship between features and final_score is largely linear and well captured by a regression line.

[77]:
```
# Introducing new features
df['avg_score'] = (df['math_score'] + df['english_score'])/2
df['score_diff'] = df['math_score'] - df['english_score']
num_cols_new = ['avg_score', 'score_diff']

# New training and test sets
X_new = df.drop(columns=['student_id', 'final_score'])
y_new = df['final_score']

X_train_new, X_test_new, y_train_new, y_test_new = train_test_split(X_new,␣
 ↪y_new, test_size = 0.3, random_state = 42)

# New ColumnTransformer
preprocessor_new = ColumnTransformer([('num', StandardScaler(), num_cols_new),
                                      ('cat', OneHotEncoder(drop = 'first'),␣
 ↪cat_cols)])

# New LinearRegression pipeline
pipe_lr_new = Pipeline([('preprocessing', preprocessor_new),
                        ('model', LinearRegression())])

# New DecisionTree pipeline
pipe_tree_new = Pipeline([('preprocessing', preprocessor_new),
                          ('model', DecisionTreeRegressor(random_state=42))])

# Training new models
pipe_lr_new.fit(X_train_new, y_train_new)
pipe_tree_new.fit(X_train_new, y_train_new)

# Predicting with new models
y_pred_lr_new = pipe_lr_new.predict(X_test_new)
```

```python
y_pred_tree_new = pipe_tree_new.predict(X_test_new)

# Evaluating  new models

# Linear Regression
mae_lr_new = mean_absolute_error(y_test_new, y_pred_lr_new)
rmse_lr_new = root_mean_squared_error(y_test_new, y_pred_lr_new)
r2_lr_new = r2_score(y_test_new, y_pred_lr_new)

# Decision Tree
mae_tree_new = mean_absolute_error(y_test_new, y_pred_tree_new)
rmse_tree_new = root_mean_squared_error(y_test_new, y_pred_tree_new)
r2_tree_new = r2_score(y_test_new, y_pred_tree_new)

#
print("Linear Regression (new features)")
print("MAE:", mae_lr_new)
print("RMSE:", rmse_lr_new)
print("R²:", r2_lr_new)

print("\nDecision Tree (new features)")
print("MAE:", mae_tree_new)
print("RMSE:", rmse_tree_new)
print("R²:", r2_tree_new)
```

```
Linear Regression (new features)
MAE: 1.246456196966804
RMSE: 1.4534331865384358
R²: 0.9875411453148221

Decision Tree (new features)
MAE: 4.666666666666667
RMSE: 5.0990195135927845
R²: 0.8466579292267365
```

After testing an alternative feature design using avg_score and score_diff, we found that the models performance remained unchanged. This is expected since these derived features are linear transformations of the original inputs (math_score, english_score) and do not provide additional predictive power. The linear model remains highly effective, with R^2 0.99.