

Feature Selection Methods for Regression Models

Lecture 10

Prepared by

Dr. Hager Saleh

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score
from sklearn.preprocessing import StandardScaler
import seaborn as sns
```

Reading dataset

```
data = pd.read_csv('final_clean_outliers.csv')
data.head()
```

	Average Outflow	Average Inflow	Energy Consumption	Ammonia \
0	2.928	3.225	202016.0	42.0
1	2.928	3.354	207547.0	36.0
2	2.917	3.794	202824.0	46.0
3	2.912	3.750	247691.0	40.0
4	2.355	3.507	224706.0	51.0

	Biological Oxygen Demand	Chemical Oxygen Demand	Total Nitrogen \
0	418.0	836.0	64.522

1	430.0	850.0	63.000
2	508.0	1016.0	65.590
3	410.0	820.0	61.786
4	555.0	1110.0	68.820

	Average Temperature	Maximum temperature	Minimum temperature \
0	16.8	27.2	8.8
1	14.6	19.9	11.1
2	13.4	19.1	8.0
3	18.9	27.3	7.7
4	25.9	32.4	12.1

	Atmospheric pressure	Average humidity	Total rainfall	Average visibility \
0	0	47.0	0.25	10.0
1	0	49.0	0.00	10.0
2	0	65.0	0.00	10.0
3	0	52.0	0.00	0.0
4	0	39.0	0.00	0.0

	Average wind speed	Maximum wind speed	Year	Month	Day
0	31.9	61.1	2014	1	5
1	27.0	38.9	2014	1	6
2	20.6	35.2	2014	1	7
3	15.6	31.7	2014	1	8
4	26.7	46.5	2014	1	9

```
X = data.drop('Energy Consumption', axis=1)
y = data['Energy Consumption']
```

Split dataset into train and test sets

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

Feature Selection Methods

Tree-Based Model Feature Importance

```
model = RandomForestRegressor()
model.fit(X_train, y_train)

# Get feature importances
feature_importances = model.feature_importances_
selected_features = X.columns[np.argsort(feature_importances)[-10:]]
```

```
# Select top 5 features
```

```
print(f"Selected features using tree-based importance:  
{selected_features.tolist()}")
```

```
Selected features using tree-based importance: ['Biological Oxygen  
Demand', 'Average humidity', 'Day', 'Chemical Oxygen Demand', 'Average  
wind speed', 'Average Outflow', 'Average Inflow', 'Year', 'Total  
Nitrogen', 'Month']
```

```
X_train_selected = X_train[selected_features]  
X_test_selected = X_test[selected_features]
```

```
scaler = StandardScaler()  
X_train_selected = scaler.fit_transform(X_train_selected)  
X_test_selected = scaler.transform(X_test_selected)
```

Applying DecisionTreeRegressor on selected features

```
def evaluate_model(model_name, y_true, y_pred):
```

```
    # Calculate metrics
```

```
    mae = mean_absolute_error(y_true, y_pred)  
    mse = mean_squared_error(y_true, y_pred)  
    rmse = np.sqrt(mse)  
    r2 = r2_score(y_true, y_pred)
```

```
    # Print the evaluation results
```

```
    results = {  
        "Mean Absolute Error (MAE)": mae,  
        "Mean Squared Error (MSE)": mse,  
        "Root Mean Squared Error (RMSE)": rmse,  
        "R^2 Score": r2  
    }
```

```
    return results
```

DecisionTreeRegressor

```
DT = DecisionTreeRegressor(random_state=42)
```

```
# Define the parameter grid
```

```
param_grid = {  
    'max_depth': [None, 10, 20, 30],  
    'min_samples_split': [2, 5, 10],  
    'min_samples_leaf': [1, 2, 4],  
}
```

```
# Initialize GridSearchCV
```

```
grid_search = GridSearchCV(
```

```

    estimator=DT,
    param_grid=param_grid,
    cv=5, # 5-fold cross-validation
    scoring='neg_root_mean_squared_error', # Metric to optimize
    verbose=2,
    n_jobs=-1 # Use all available cores
)

# Fit the grid search
grid_search.fit(X_train_selected, y_train)

# Get the best parameters and the corresponding score
print("Best Parameters:", grid_search.best_params_)
print("Best CV RMSE:", grid_search.best_score_)

# Train a new model using the best parameters
best_DT = grid_search.best_estimator_

# Evaluate the model on the test set
y_pred = best_DT.predict(X_test_selected)
evaluation_results = evaluate_model('Decision Tree Regressor', y_test,
y_pred)

# Print the evaluation results
print("\nEvaluation Results on Test Set:")
for key, value in evaluation_results.items():
    print(f"{key}: {value:.4f}" if isinstance(value, float) else
f"{key}: \n{value}")

Fitting 5 folds for each of 36 candidates, totalling 180 fits
Best Parameters: {'max_depth': 10, 'min_samples_leaf': 4,
'min_samples_split': 10}
Best CV RMSE: -35962.4877262347

Evaluation Results on Test Set:
Mean Absolute Error (MAE): 24996.4825
Mean Squared Error (MSE): 1034147885.4865
Root Mean Squared Error (RMSE): 32158.1698
R^2 Score: 0.1172

```

Recursive Feature Elimination - RFE

```

from sklearn.feature_selection import RFE
from sklearn.linear_model import LinearRegression

# Create a base model
model = LinearRegression()

# Use RFE for feature selection
rfe = RFE(model, n_features_to_select=8)

```

```

X_rfe = rfe.fit_transform(X, y)

# Get selected feature names
selected_features = X.columns[rfe.support_]
print(f"Selected features using RFE: {selected_features.tolist()}")

Selected features using RFE: ['Average Outflow', 'Ammonia', 'Total Nitrogen', 'Average Temperature', 'Maximum temperature', 'Total rainfall', 'Year', 'Month']

X_train_selected = X_train[selected_features]
X_test_selected = X_test[selected_features]

scaler = StandardScaler()
X_train_selected = scaler.fit_transform(X_train_selected)
X_test_selected = scaler.transform(X_test_selected)

```

Applying RandomForestRegressor on selected features

```

rf_regressor = RandomForestRegressor()

# Define the parameter grid
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
}

# Initialize GridSearchCV
grid_search = GridSearchCV(
    estimator=rf_regressor,
    param_grid=param_grid,
    cv=5, # 5-fold cross-validation
    scoring='neg_root_mean_squared_error', # Metric to optimize
    verbose=2,
    n_jobs=-1 # Use all available cores
)

# Fit the grid search
grid_search.fit(X_train_selected, y_train)

# Get the best parameters and the corresponding score
print("Best Parameters:", grid_search.best_params_)

# Train a new model using the best parameters
best_rf = grid_search.best_estimator_

# Evaluate on the test set
y_pred = best_rf.predict(X_test_selected)

```

```
evaluation_results = evaluate_model('RandomForestRegressor', y_test,
y_pred)
```

```
# Print the evaluation results
```

```
print("\nEvaluation Results on Test Set:")
```

```
for key, value in evaluation_results.items():
```

```
    print(f"{key}: {value:.4f}" if isinstance(value, float) else
f"{key}: \n{value}")
```

Fitting 5 folds for each of 108 candidates, totalling 540 fits

Best Parameters: {'max_depth': 20, 'min_samples_leaf': 2,
'min_samples_split': 5, 'n_estimators': 200}

Evaluation Results on Test Set:

Mean Absolute Error (MAE): 23289.1487

Mean Squared Error (MSE): 856465827.7483

Root Mean Squared Error (RMSE): 29265.4374

R² Score: 0.2689