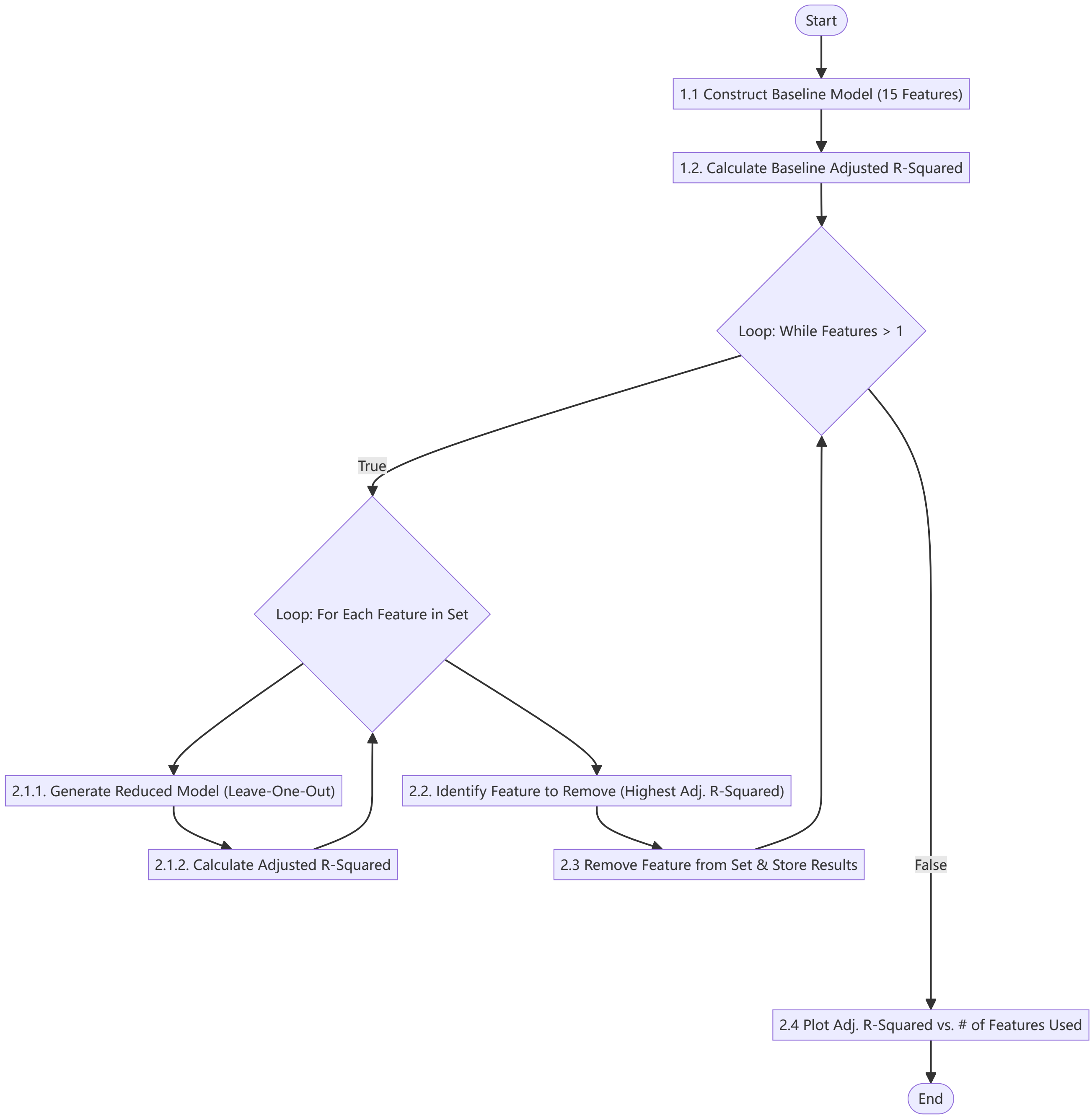
CS 439 25F DataBench Bonus Assignment

Multi-Linear Regression Model of Student Assessment Performance Questionnaires

by James Abello, Haoyang Zhang



Dataset Description: Student Assessment Questionnaires and Quiz Scores

The dataset `assessment_quiz_generated.csv` contains information derived from student assessment questionnaires and their corresponding quiz scores.

The dataset comprises the following attributes:

- `timestamp`
  The date and time when the assessment was submitted, formatted as `yyyy-mm-dd hh:mm:ss timezone`.

- `netid`
  The encoded NetID of the student. Valid NetIDs must have a string length between 8 and 14 characters (inclusive). Entries falling outside this range are considered invalid.

- `ruid`
  The encoded RUID of the student. A valid RUID is expected to contain exactly 18 characters. Any deviation from this length is considered invalid.

- **Skill Proficiency Columns**
  The following columns record students' self-assessed proficiency levels in specific skills, rated on scales ranging from 0 up to a multiple of 5 (depending on the number of questions per topic).

  - 0: `data_structures`
  - 1: `calculus_and_linear_algebra`
  - 2: `probability_and_statistics`
  - 3: `data_visualization`
  - 4: `python_libraries`
  - 5: `shell_scripting`
  - 6: `sql`
  - 7: `python_scripting`
  - 8: `jupyter_notebook`
  - 9: `regression`
  - 10: `programming_languages`
  - 11: `algorithms`
  - 12: `complexity_measures`
  - 13: `visualization_tools`
  - 14: `massive_data_processing`
- `quiz_score` **(New Added column to the previous dataset)** The score obtained by the student in the quiz, represented as a floating-point number between 0 and 100 (inclusive).

Assessment Tasks

Complete the following tasks by using the provided Jupyter Notebook Template `ME-AR.ipynb` in the folder "Model Evaluation".

Part 1: Baseline Model Construction

1. **Build a Multiple Linear Regression Model** (Task 1.1): Construct a multiple linear regression model using `train_linear_regression_model` to predict `quiz_score` using all 15 skill proficiency columns as predictor variables.
2. **Evaluate Model Performance using Adjusted R-Squared** (Task 1.2): Evaluate the performance of the baseline model by calculating and reporting its adjusted R-squared value. The formula for adjusted R-squared is:

$$\text{Adjusted } R^2 = 1 - (1 - R^2)\frac{n-1}{n-p-1}$$

where n is the number of observations and p is the number of predictors.

Part 2: Feature Importance Analysis

1. **Candidate Model Generation and Adjusted R-Squared Calculation** (Task 2.1): Systematically evaluate the importance of each feature by performing the following steps for each of the 15 skill proficiency columns:
   - a. A collection of reduced models is generated by systematically leaving out one feature at a time from the current feature set.
   - b. The adjusted R-squared is computed for each reduced model.
2. **Rank Features by Importance** (Task 2.2): The feature whose removal results in the highest adjusted R-squared is identified as the least important feature. The selected least important feature is permanently removed from the set of predictors.
3. **Iterative Refinement** (Task 2.3): Repeat the above process iteratively, each time removing the least important feature from the current set of predictors, until only one feature is left.
4. **Performance Visualization** (Task 2.4): The relationship between model complexity and performance is visualized by plotting the adjusted R-squared against the number of features used at each iteration. Each point on the plot is annotated with the index of the feature that was removed at that stage.

Environment Setup and Data Loading

In [ ]:
```python
from sklearn.linear_model import LinearRegression
import pandas as pd
from matplotlib import pyplot as plt
import numpy as np
```

In [2]:
```python
def load_data(file_path):
    """Load the dataset from a CSV file.
    IN: file_path: str, path to the CSV file
    OUT: pd.DataFrame, loaded dataset
    """
    return pd.read_csv(file_path)
```

In [3]:
```python
if __name__ == "__main__":
    skill_columns = [
        'data_structures',
        'calculus_and_linear_algebra',
        'probability_and_statistics',
        'data_visualization',
        'python_libraries',
        'shell_scripting',
        'sql',
        'python_scripting',
        'jupyter_notebook',
        'regression',
        'programming_languages',
        'algorithms',
        'complexity_measures',
        'visualization_tools',
        'massive_data_processing'
    ]

    # Load the dataset
```

```
    data = load_data('assessment_quiz_generated.csv')

    display(data.describe())
    display(data.info())
```

	data_structures	calculus_and_linear_algebra	probability_and_statistics	data_visualization	python_l
count	105.000000	105.000000	105.000000	105.000000	105
mean	20.466667	14.323810	31.057143	22.000000	8
std	8.882856	4.878318	13.724507	8.212327	5
min	0.000000	0.000000	0.000000	0.000000	0
25%	16.000000	11.000000	24.000000	16.000000	2
50%	22.000000	15.000000	31.000000	22.000000	9
75%	26.000000	18.000000	41.000000	28.000000	12
max	35.000000	25.000000	55.000000	35.000000	20

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 105 entries, 0 to 104
Data columns (total 19 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   timestamp                    105 non-null    object
 1   netid                        105 non-null    object
 2   ruid                         105 non-null    object
 3   data_structures              105 non-null    int64
 4   calculus_and_linear_algebra  105 non-null    int64
 5   probability_and_statistics   105 non-null    int64
 6   data_visualization           105 non-null    int64
 7   python_libraries             105 non-null    int64
 8   shell_scripting              105 non-null    int64
 9   sql                          105 non-null    int64
 10  python_scripting             105 non-null    int64
 11  jupyter_notebook             105 non-null    int64
 12  regression                   105 non-null    int64
 13  programming_languages        105 non-null    int64
 14  algorithms                   105 non-null    int64
 15  complexity_measures          105 non-null    int64
 16  visualization_tools          105 non-null    int64
 17  massive_data_processing      105 non-null    int64
 18  quiz_score                   105 non-null    float64
dtypes: float64(1), int64(15), object(3)
memory usage: 15.7+ KB
None
```

Task 1.1: Build a Muti-Linear Regression Model

In [ ]:

```
def train_linear_regression_model(df, feature_cols, target_col):
    """Train a linear regression model.
    IN: df: pd.DataFrame, the dataset
        feature_cols: list of str, names of feature columns
        target_col: str, name of the target column
    OUT: LinearRegression, trained model
    """
    # Your code here
    return model
```

In [5]:
```python
if __name__ == "__main__":
    # train the model with all features
    model = train_linear_regression_model(data, skill_columns, 'quiz_score')
```

Task 1.2: Evaluate Model Performance using Adjusted R-Squared

In [ ]:
```python
def calculate_adjusted_r_squared(model, df, feature_cols, target_col):
    """Calculate the adjusted R-squared of the model.
    IN: model: LinearRegression, trained model
        df: pd.DataFrame, the dataset
        feature_cols: list of str, names of feature columns
        target_col: str, name of the target column
    OUT: float, adjusted R-squared value
    """
    # Your code here
    return adjusted_r_squared
```

In [7]:
```python
if __name__ == "__main__":
    adjusted_r2 = calculate_adjusted_r_squared(model, data, skill_columns, 'quiz_score')
    print(f'Adjusted R-squared: {adjusted_r2}')
```

```
Adjusted R-squared: 0.5308076296068789
```

Task 2: Feature Importance Analysis

2.1. Candidate Model Generation and Adjusted R-Squared Calculation

In [ ]:
```python
def calculate_reduced_model_adjusted_r_squared(df, feature_cols, target_col, remove_col):
    """Calculate adjusted R-squared after removing one feature.
    IN: df: pd.DataFrame, the dataset
        feature_cols: list of str, names of current feature columns
        target_col: str, name of the target column
        remove_col: str, name of the feature to remove
    OUT: float, adjusted R-squared value of the reduced model
    """
    # Your code here
    return adjusted_r2
```

In [9]:
```python
if __name__ == "__main__":
    for col in skill_columns:
        reduced_adj_r2 = calculate_reduced_model_adjusted_r_squared(data, skill_columns, 'quiz_scor
        print(f'Removed {col}, Adjusted R-squared: {reduced_adj_r2}')
```

```
Removed data_structures, Adjusted R-squared: 0.533684639029099
Removed calculus_and_linear_algebra, Adjusted R-squared: 0.5171839094369506
Removed probability_and_statistics, Adjusted R-squared: 0.48761090000458307
Removed data_visualization, Adjusted R-squared: 0.5007412533169568
Removed python_libraries, Adjusted R-squared: 0.5356859667004188
Removed shell_scripting, Adjusted R-squared: 0.5323024432710916
Removed sql, Adjusted R-squared: 0.5358517474021773
Removed python_scripting, Adjusted R-squared: 0.49599675509212326
Removed jupyter_notebook, Adjusted R-squared: 0.5359214738307233
Removed regression, Adjusted R-squared: 0.5106839440525477
Removed programming_languages, Adjusted R-squared: 0.5359945633832968
Removed algorithms, Adjusted R-squared: 0.5248976264163654
Removed complexity_measures, Adjusted R-squared: 0.5350053803049647
Removed visualization_tools, Adjusted R-squared: 0.5357509929424005
Removed massive_data_processing, Adjusted R-squared: 0.5353572283671284
```

2.2. Rank Features by Importance

In [ ]:

```python
def reduce_model(df, feature_cols, target_col):
    """Remove the least important feature based on adjusted R-squared.
    IN: df: pd.DataFrame, the dataset
        feature_cols: list of str, names of current feature columns
        target_col: str, name of the target column
    OUT: remaining_cols, list of str, updated feature columns after removal
         removed_col, str, name of the removed feature
         adjusted_r2, float, adjusted R-squared of the reduced model
    """
    # Your code here
    return remaining_cols, removed_col, best_adj_r2
```

In [11]:

```python
if __name__ == "__main__":
    remaining_cols, removed_col, adjusted_r2 = reduce_model(data, skill_columns, 'quiz_score')
    print(f'Removed feature: {removed_col}')
    print(f'Adjusted R-squared: {adjusted_r2}')
```

```
Removed feature: programming_languages
Adjusted R-squared: 0.5359945633832968
```

Subtask 2.3: Iterative Refinement

Repeat the process of removing the least important feature (as identified in Subtask 2.2) until only one feature remains. After each removal, retrain the model and record the adjusted R-squared value.

In [ ]:

```python
def feature_importance_ranking(df, feature_cols, target_col):
    """Iteratively remove the least important feature and rank features by importance.
    IN: df: pd.DataFrame, the dataset
        feature_cols: list of str, names of current feature columns
        target_col: str, name of the target column
    OUT: list of tuples (feature_name, adjusted_r_squared) in order of removal
    """
    ranking = []
    current_features = feature_cols.copy()

    full_model = train_linear_regression_model(df, current_features, target_col)
    full_adjusted_r2 = calculate_adjusted_r_squared(full_model, df, current_features, target_col)
    ranking.append(('Full Model', full_adjusted_r2))

    # Your code here

    ranking.append((current_features[0], 0))  # Last remaining feature

    return ranking
```

In [ ]:

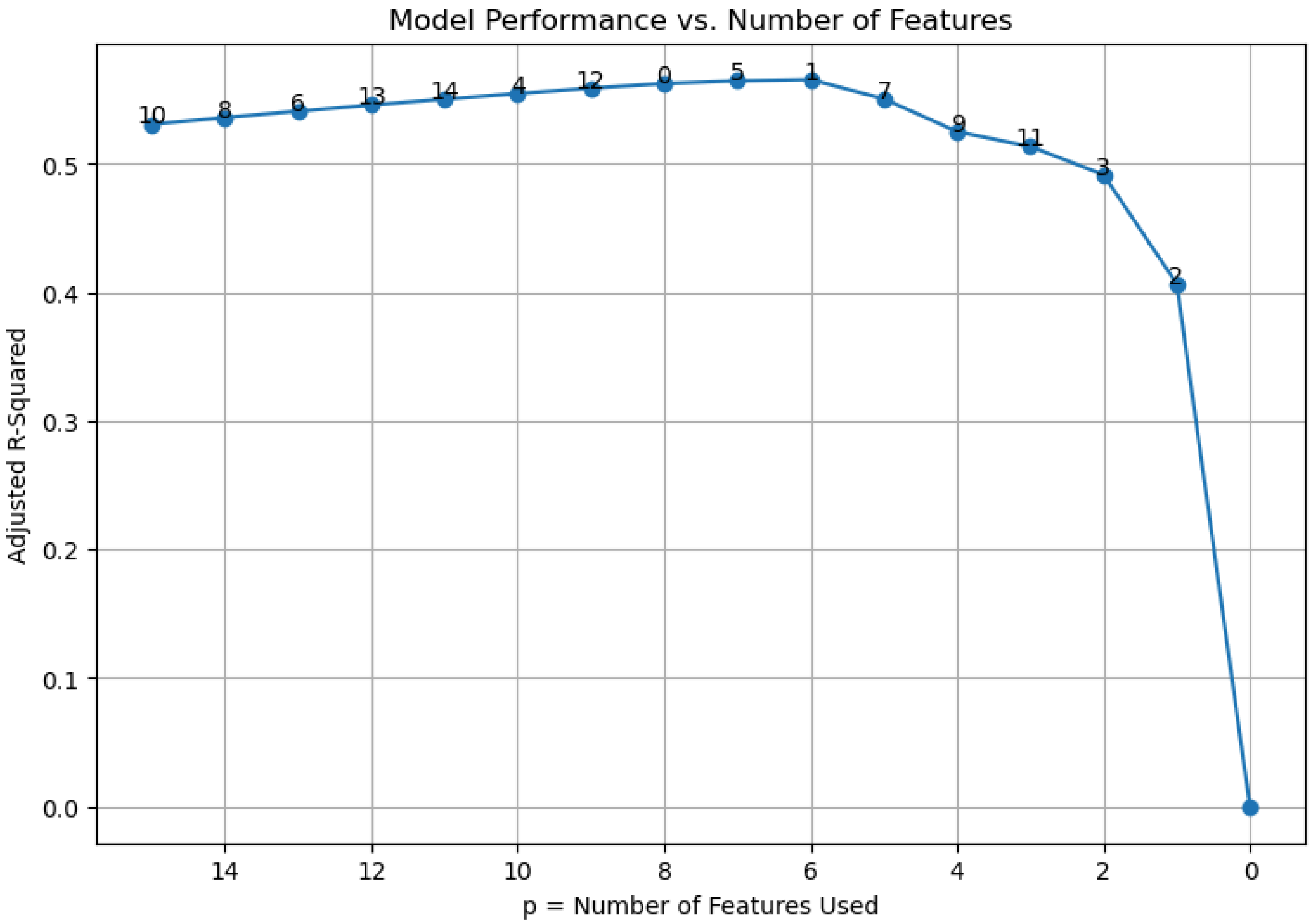
```python
if __name__ == "__main__":
    ranking = feature_importance_ranking(data, skill_columns, 'quiz_score')
    for idx, (feature, adj_r2) in enumerate(ranking):
        rank = len(ranking) - idx
        print(f' Rank: {rank:<2}, Adjusted R-squared: {adj_r2:<.4f}, Feature Removed: {feature}')
```

```
Rank: 16, Adjusted R-squared: 0.5308, Feature Removed: None
Rank: 15, Adjusted R-squared: 0.5360, Feature Removed: programming_languages
Rank: 14, Adjusted R-squared: 0.5410, Feature Removed: jupyter_notebook
Rank: 13, Adjusted R-squared: 0.5458, Feature Removed: sql
Rank: 12, Adjusted R-squared: 0.5503, Feature Removed: visualization_tools
Rank: 11, Adjusted R-squared: 0.5547, Feature Removed: massive_data_processing
Rank: 10, Adjusted R-squared: 0.5589, Feature Removed: python_libraries
Rank: 9 , Adjusted R-squared: 0.5624, Feature Removed: complexity_measures
Rank: 8 , Adjusted R-squared: 0.5646, Feature Removed: data_structures
Rank: 7 , Adjusted R-squared: 0.5656, Feature Removed: shell_scripting
Rank: 6 , Adjusted R-squared: 0.5506, Feature Removed: calculus_and_linear_algebra
Rank: 5 , Adjusted R-squared: 0.5251, Feature Removed: python_scripting
Rank: 4 , Adjusted R-squared: 0.5135, Feature Removed: regression
Rank: 3 , Adjusted R-squared: 0.4915, Feature Removed: algorithms
Rank: 2 , Adjusted R-squared: 0.4067, Feature Removed: data_visualization
Rank: 1 , Adjusted R-squared: 0.0000, Feature Removed: probability_and_statistics
```

Subtask 2.4 Performance Visualization

In [ ]:
```python
def plot_adjusted_r_2(ranking, skill_columns):
    """
    Draw a line plot of adjusted R-squared values against the number of features.

    This function plots the adjusted R-squared value for models with a decreasing number of feature
    from a full model down to a single-feature model. It also annotates each point with the index
    of the feature that was removed to achieve the next model, indicating the least important featu

    IN: ranking, list of tuples (feature_name, adjusted_r_squared) in order of removal
        skill_columns, list of str, names of all feature columns

    OUT:
    """
    plt.figure(figsize=(9, 6))

    # Your code here


    plt.xlabel('p = Number of Features Used')
    plt.ylabel('Adjusted R-Squared')
    plt.title('Model Performance vs. Number of Features')
    plt.grid(True)
    plt.gca().invert_xaxis()  # Invert x-axis to show features being removed from left to right
    plt.show()
```

In [ ]:
```python
if __name__ == "__main__":
    plt_adjusted_r_2(ranking, skill_columns)
```



Model Performance vs. Number of Features
Adjusted R-Squared
p = Number of Features Used
0.0
0.1
0.2
0.3
0.4
0.5
14
12
10
8
6
4
2
0
10
8
6
13
14
4
12
0
5
1
7
9
11
3
2