Data Processing - Final Project

Iker Aldasoro Marculeta

Jon Lejardi Jericó

Table of Contents

1. Introduction
2. Analysis of Input Variables
3. Text Preprocessing
4. Vector representation
5. Regression Models
6. Result Comparison

Introduction

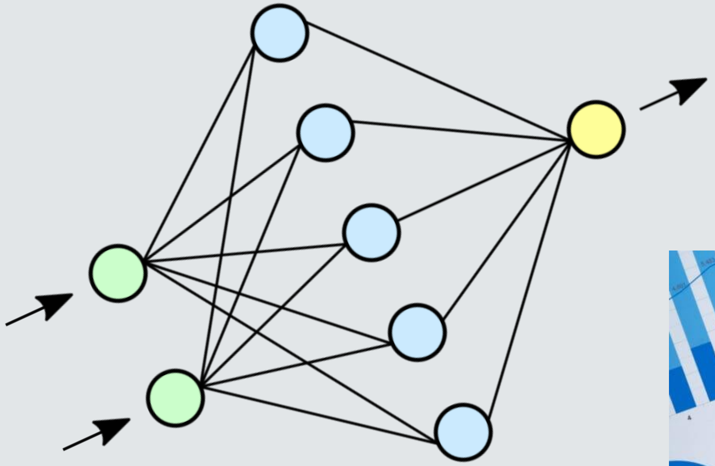
{.json}

1 output

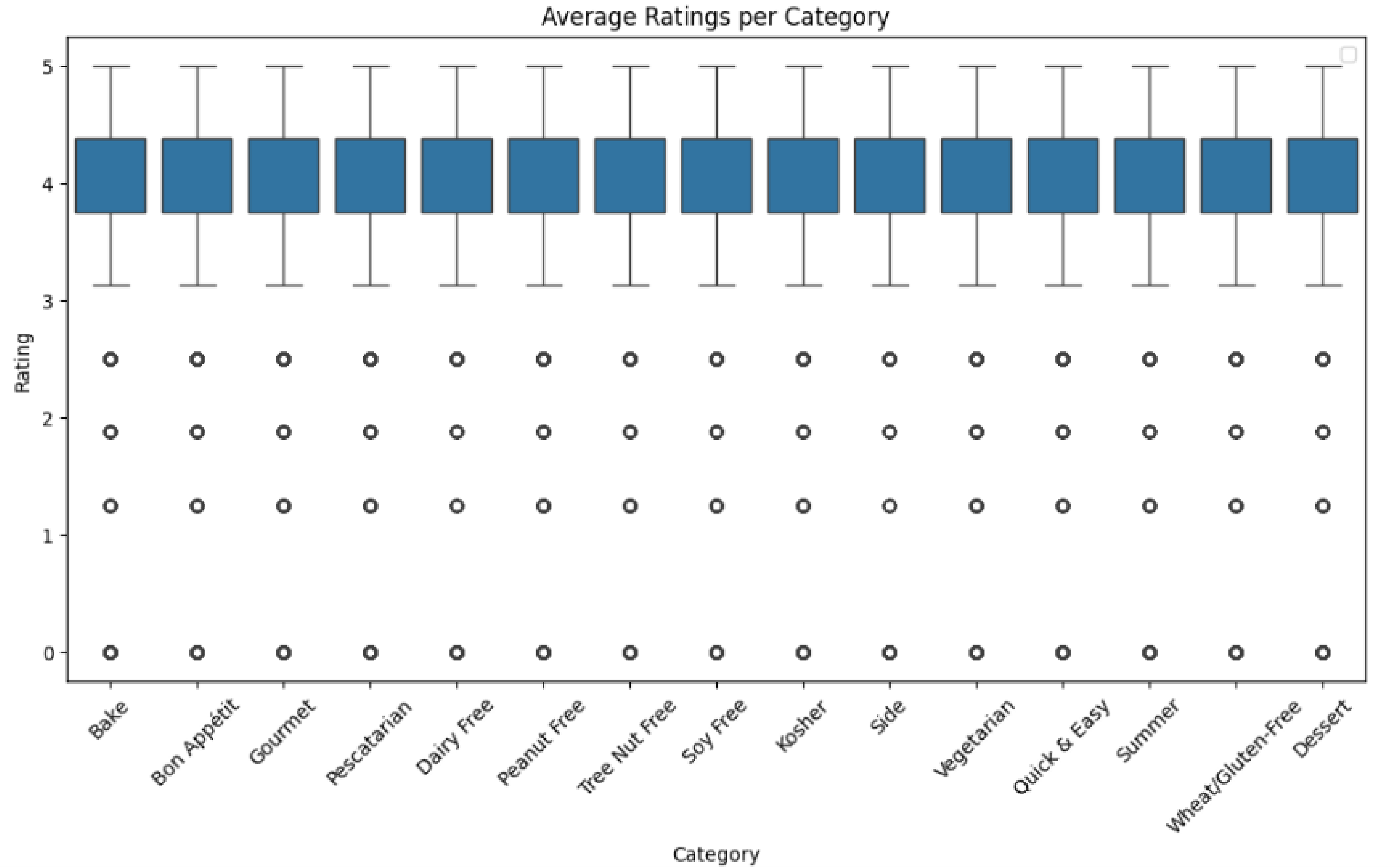
Integer variable “rating”

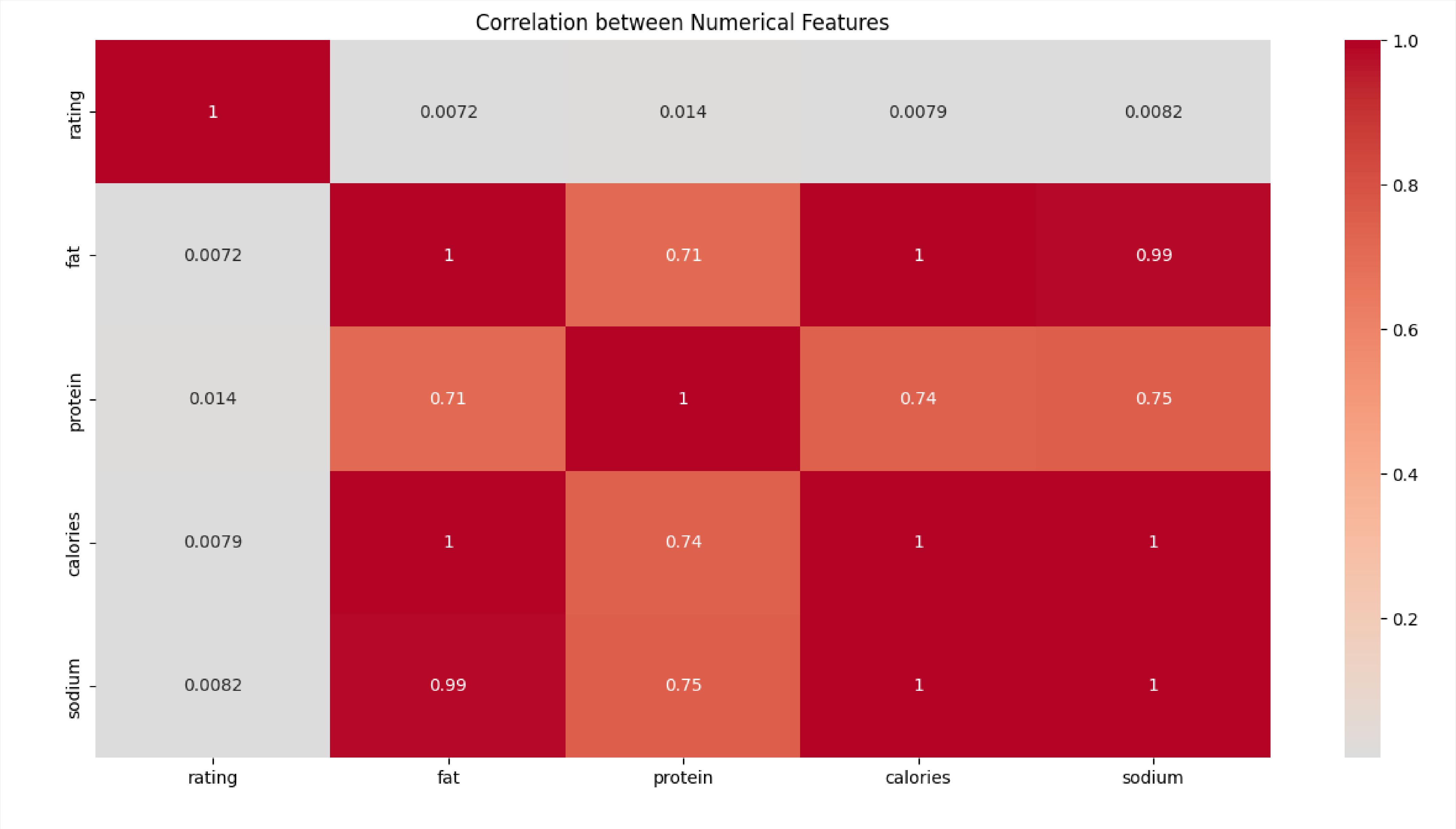
10 inputs

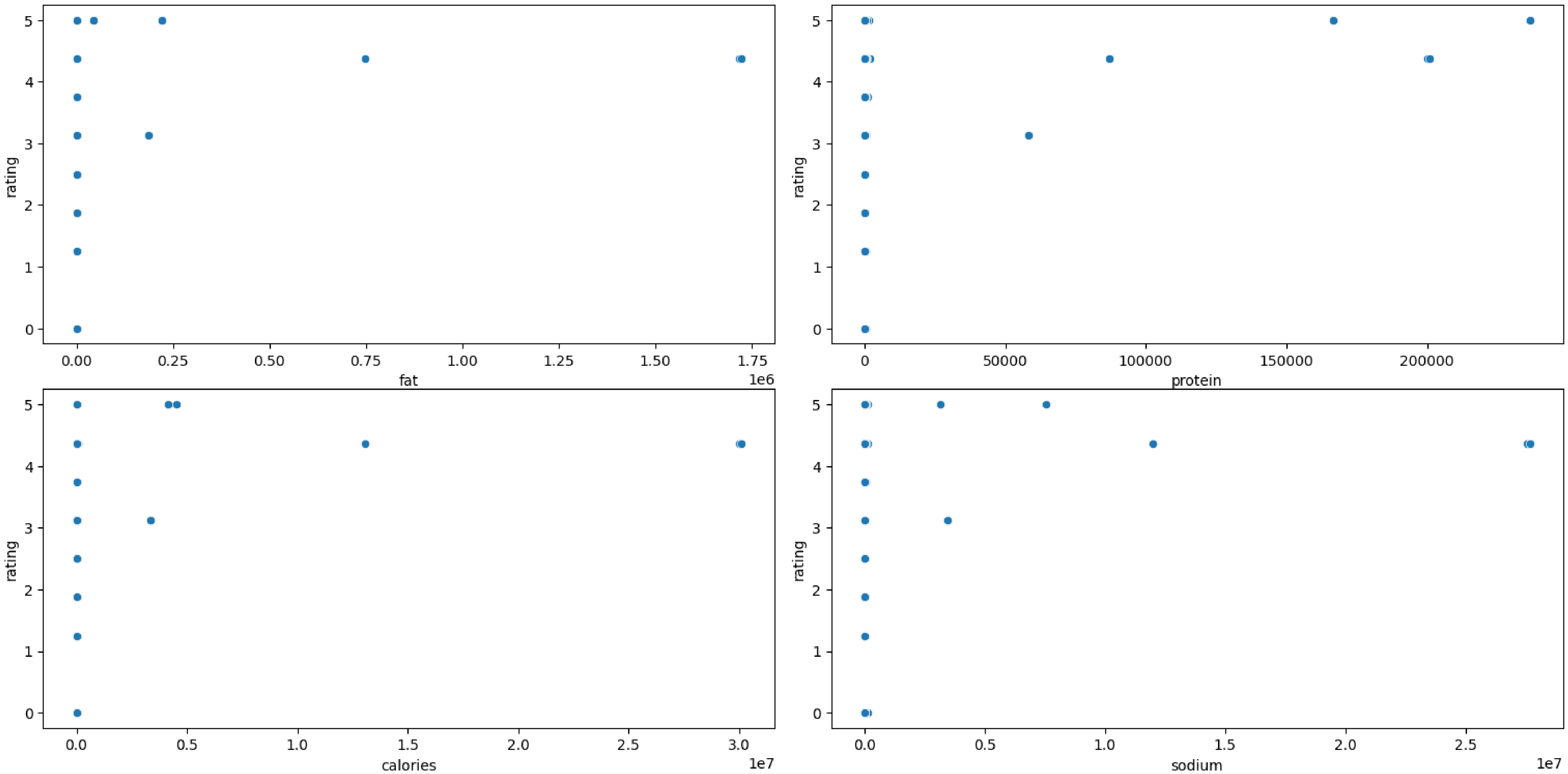
4 text variables
6 integers

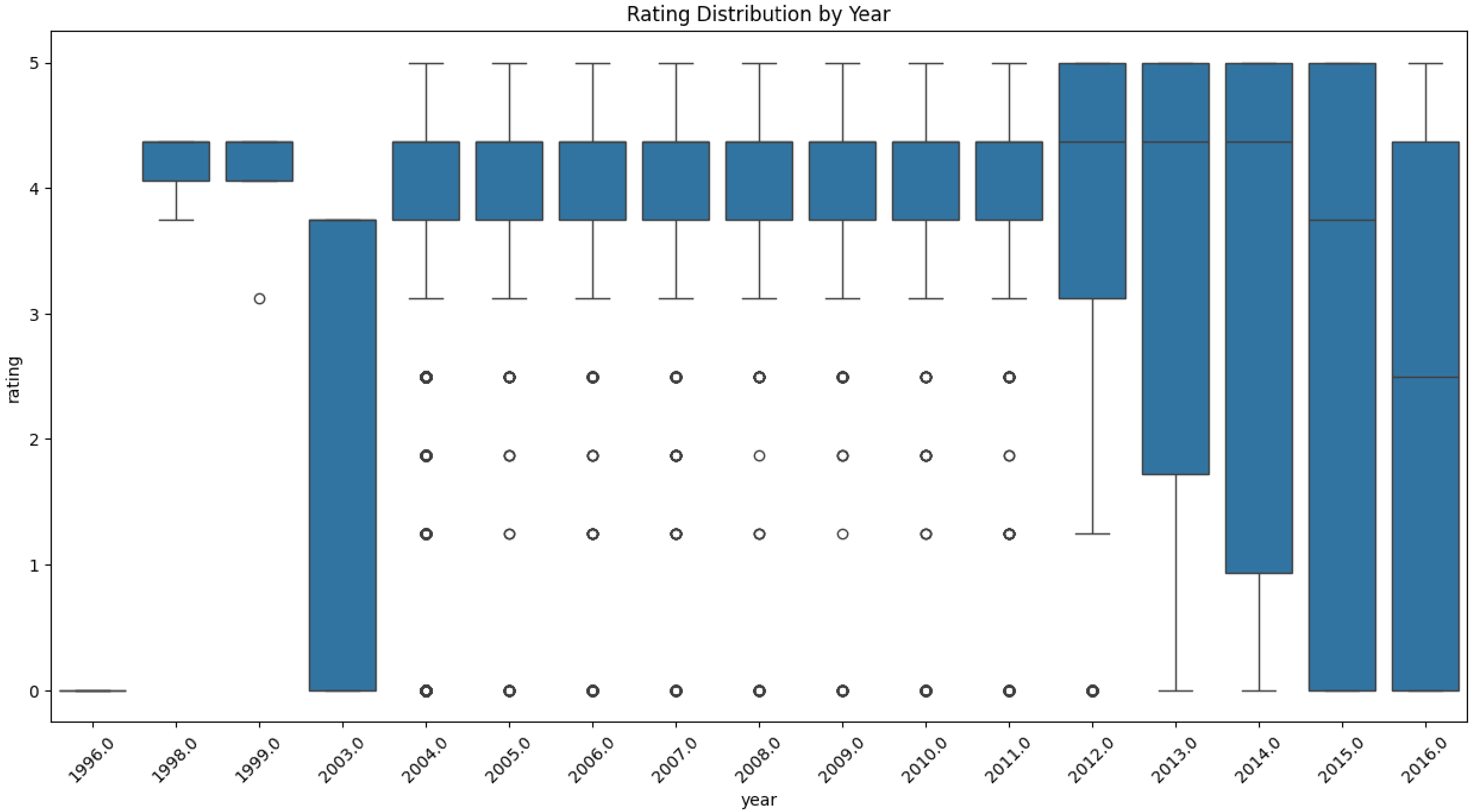


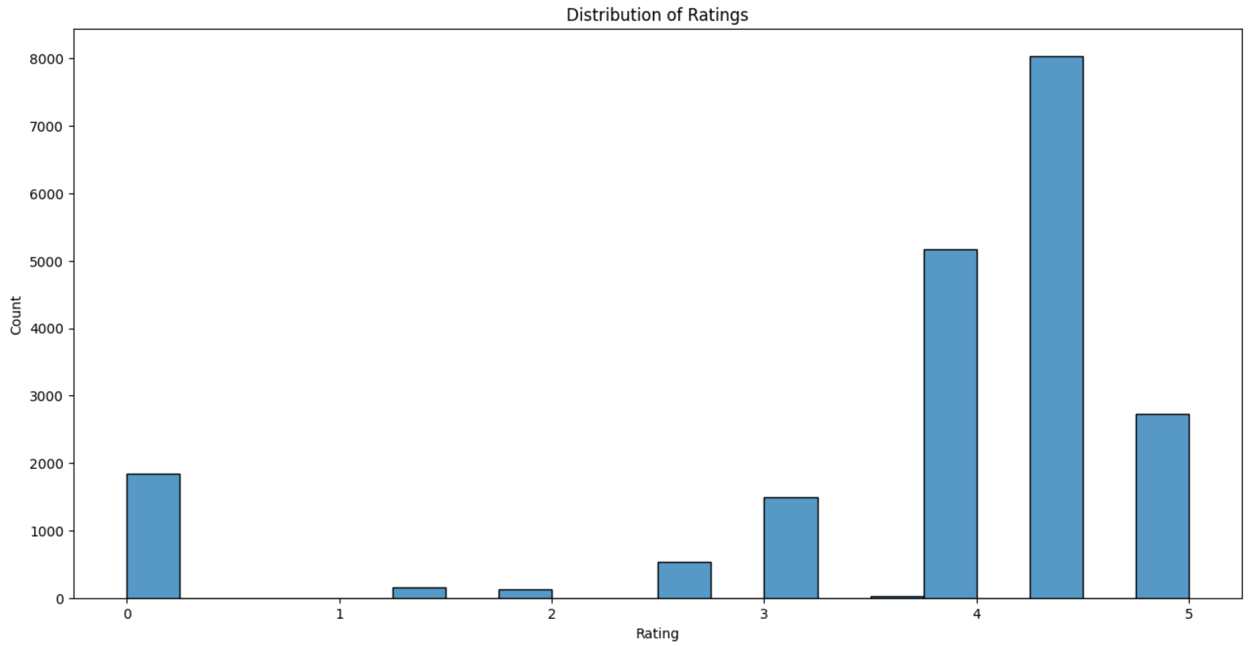
Input Variable Analysis

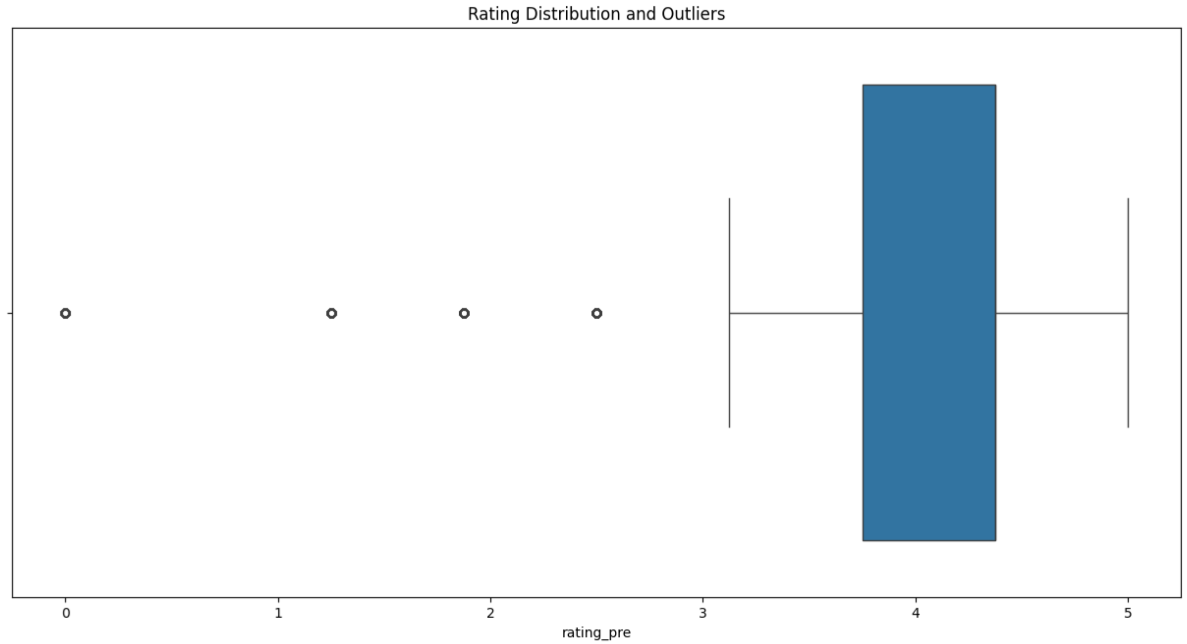

Average Ratings per Category
Rating
5
4
3
2
1
0
Category
Bake
Bon Appétit
Gourmet
Pescatarian
Dairy Free
Peanut Free
Tree Nut Free
Soy Free
Kosher
Side
Vegetarian
Quick & Easy
Summer
Wheat/Gluten-Free
Dessert


Correlation between Numerical Features
rating
fat
protein
calories
sodium
1
0.0072
0.014
0.0079
0.0082
0.0072
1
0.71
1
0.99
0.014
0.71
1
0.74
0.75
0.0079
1
0.74
1
1
0.0082
0.99
0.75
1
1
rating
fat
protein
calories
sodium
1.0
0.8
0.6
0.4
0.2

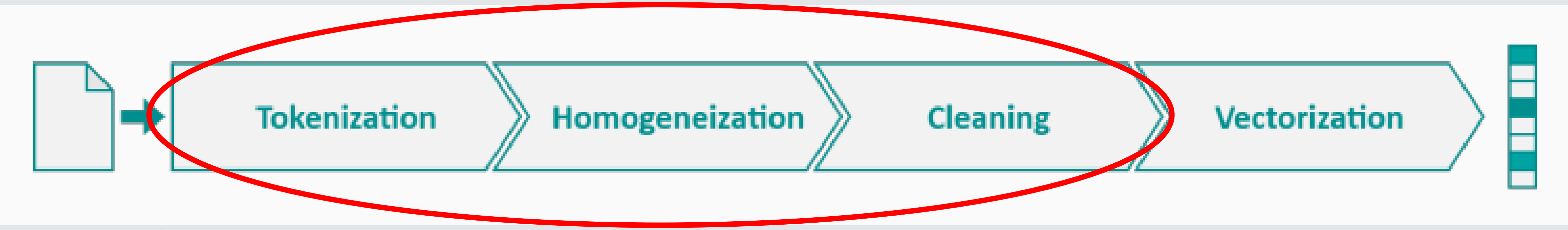

rating
0
1
2
3
4
5
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
1e6
fat
rating
0
1
2
3
4
5
0
50000
100000
150000
200000
protein
rating
0
1
2
3
4
5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
1e7
calories
rating
0
1
2
3
4
5
0.0
0.5
1.0
1.5
2.0
2.5
1e7
sodium


Rating Distribution by Year
rating
5
4
3
2
1
0
year
1996.0
1998.0
1999.0
2003.0
2004.0
2005.0
2006.0
2007.0
2008.0
2009.0
2010.0
2011.0
2012.0
2013.0
2014.0
2015.0
2016.0


Distribution of Ratings
Count
8000
7000
6000
5000
4000
3000
2000
1000
0
0
1
2
3
4
5
Rating


Rating Distribution and Outliers
0
1
2
3
4
5
rating_pre

Text Preprocessing



Tokenization

- Divide phrases into words

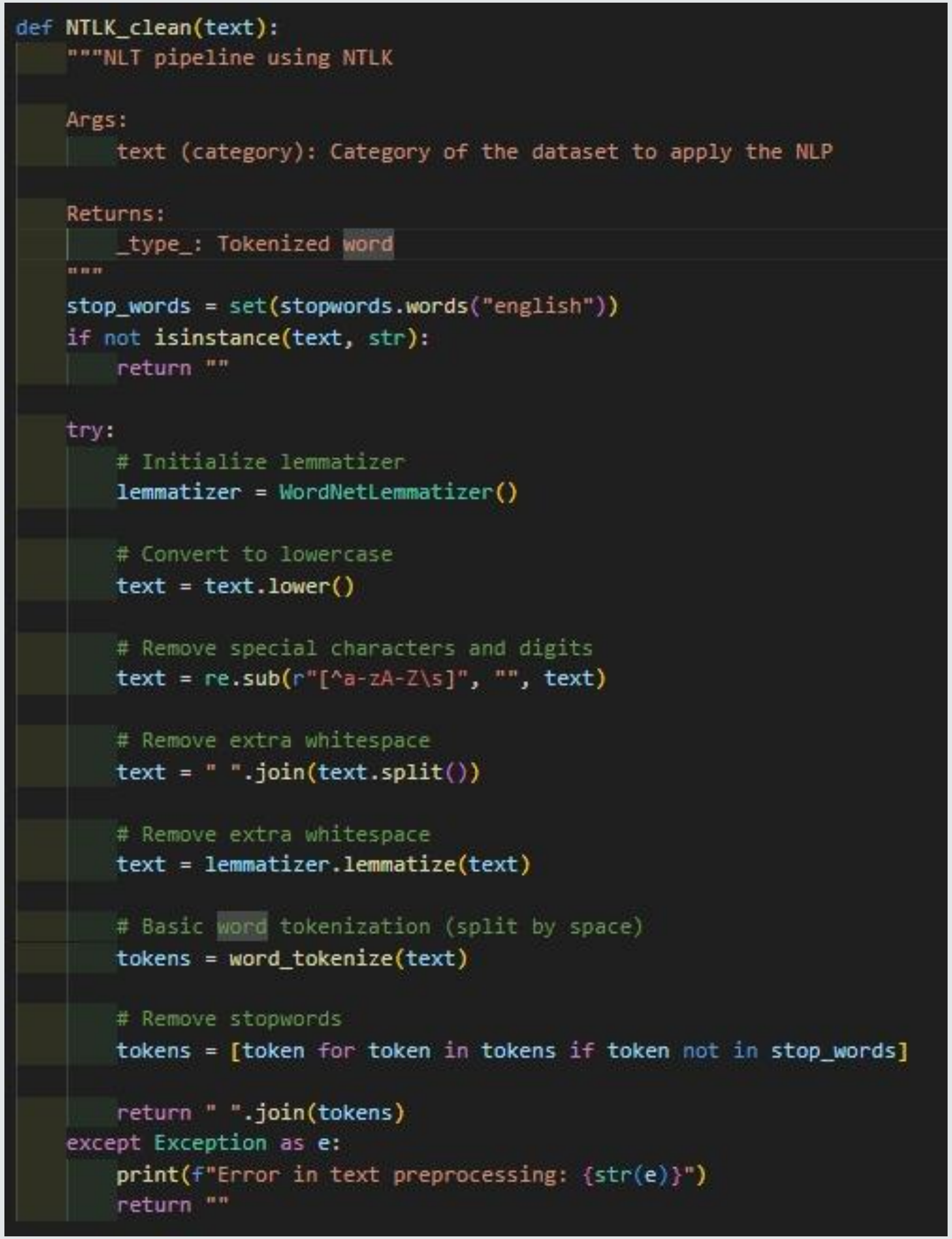
Homogeneization

- Lemmatization
- Convert to lowercase

Cleaning

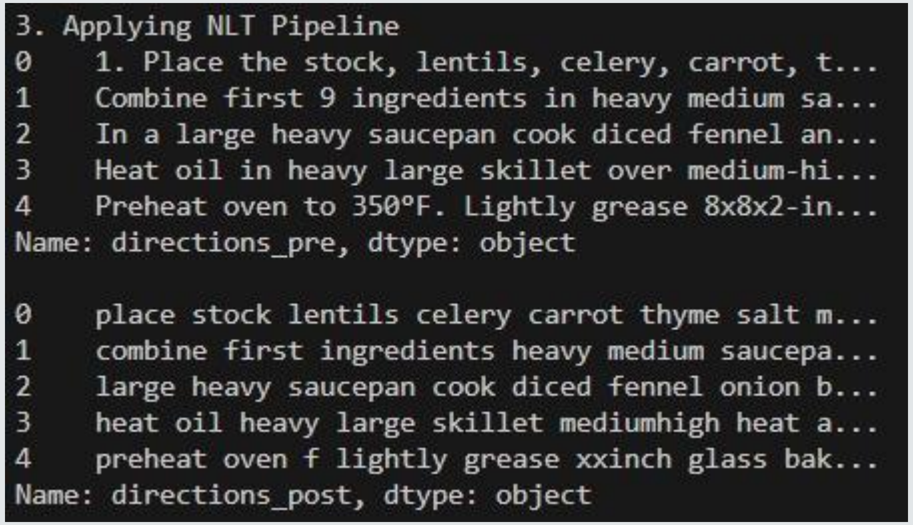
- Remove stop words, extra white spaces, special characters …

Code:



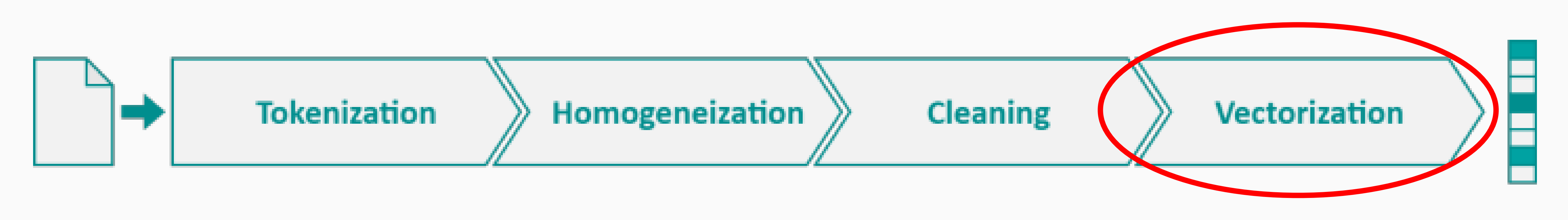
```python
def NTLK_clean(text):
    """NLT pipeline using NTLK

    Args:
        text (category): Category of the dataset to apply the NLP

    Returns:
        _type_: Tokenized word
    """
    stop_words = set(stopwords.words("english"))
    if not isinstance(text, str):
        return ""

    try:
        # Initialize lemmatizer
        lemmatizer = WordNetLemmatizer()

        # Convert to lowercase
        text = text.lower()

        # Remove special characters and digits
        text = re.sub(r"[^a-zA-Z\s]", "", text)

        # Remove extra whitespace
        text = " ".join(text.split())

        # Remove extra whitespace
        text = lemmatizer.lemmatize(text)

        # Basic word tokenization (split by space)
        tokens = word_tokenize(text)

        # Remove stopwords
        tokens = [token for token in tokens if token not in stop_words]

        return " ".join(tokens)
    except Exception as e:
        print(f"Error in text preprocessing: {str(e)}")
        return ""
```

Output:



```
3. Applying NLT Pipeline
0    1. Place the stock, lentils, celery, carrot, t...
1    Combine first 9 ingredients in heavy medium sa...
2    In a large heavy saucepan cook diced fennel an...
3    Heat oil in heavy large skillet over medium-hi...
4    Preheat oven to 350°F. Lightly grease 8x8x2-in...
Name: directions_pre, dtype: object

0    place stock lentils celery carrot thyme salt m...
1    combine first ingredients heavy medium saucepa...
2    large heavy saucepan cook diced fennel onion b...
3    heat oil heavy large skillet mediumhigh heat a...
4    preheat oven f lightly grease xxinch glass bak...
Name: directions_post, dtype: object
```

Vector Representation



Convert text data into **Word vectors** (numerical) to make inputs compatible to ML models.

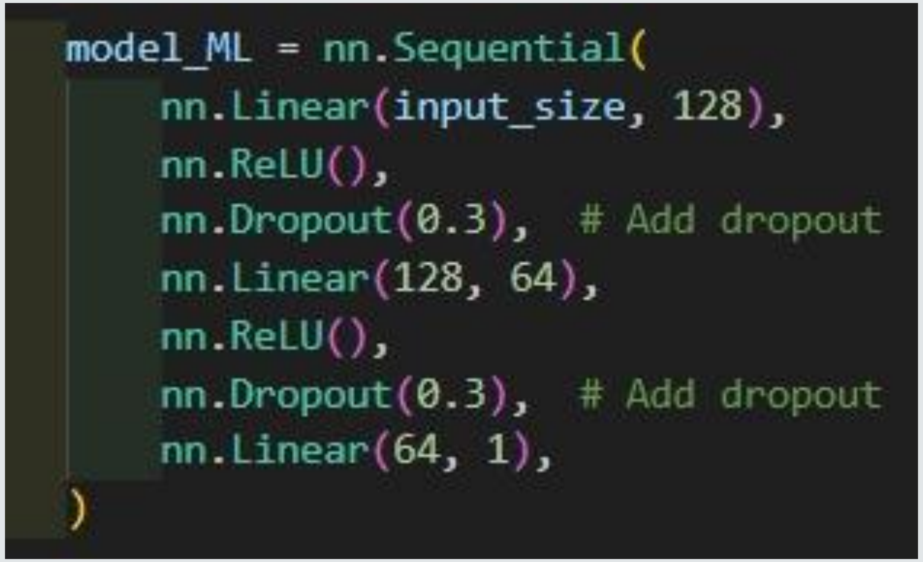
- TF-IDF: Context-agnostic and focuses on word importance (20130, 1000).
- Word2Vec: Captures shallow word relationships, lacks full contextual understanding: (20130, 100), vector size 100 and window 5. -> Mean vector
- BERT: Context-aware and provides deep contextual embeddings (20130, 768).

Regression Models

Neural Networks

Multi-Layer NN for regression

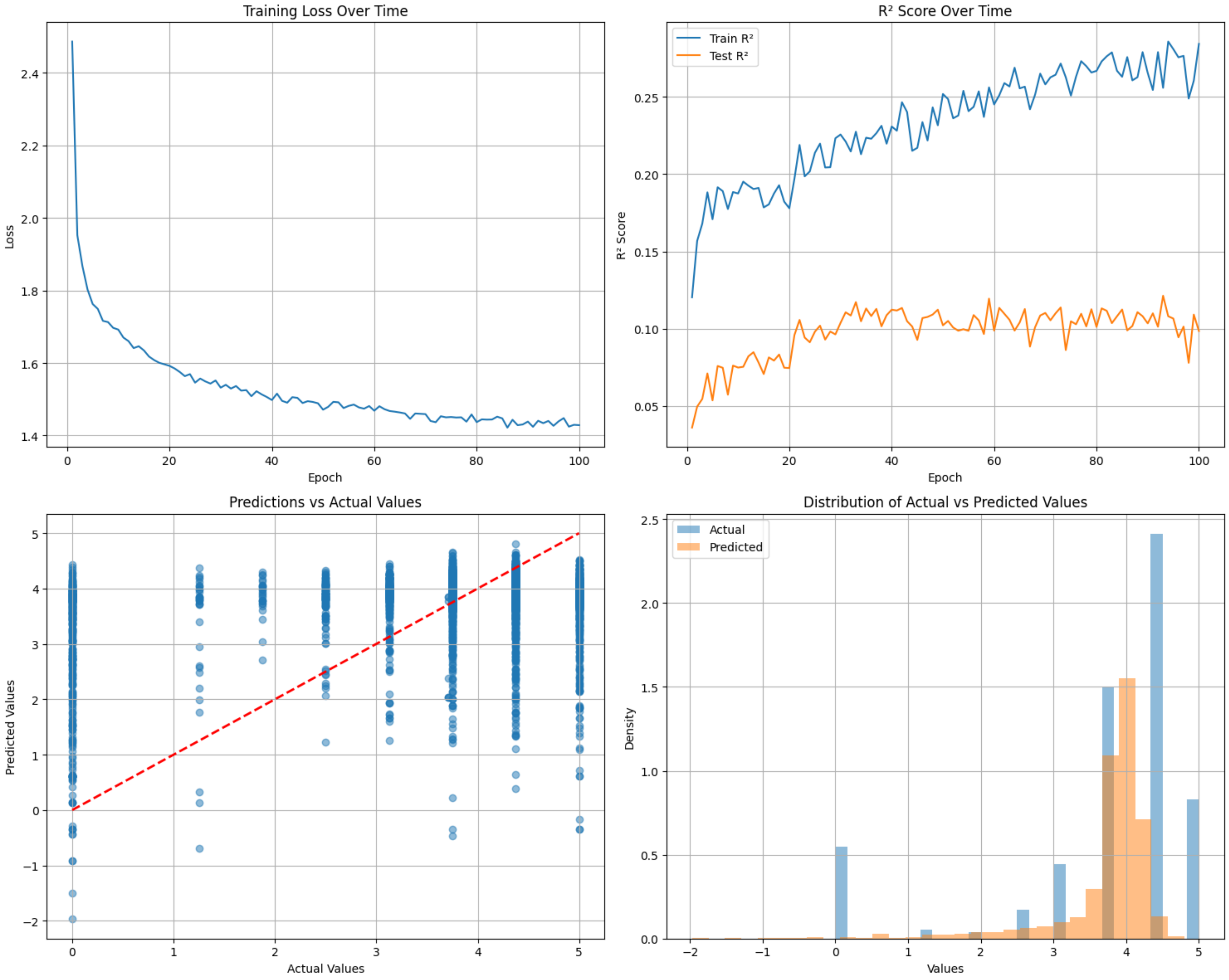
- 2 hidden layers (128 and 64 features).
- ReLu activation function.
- 30% dropout.



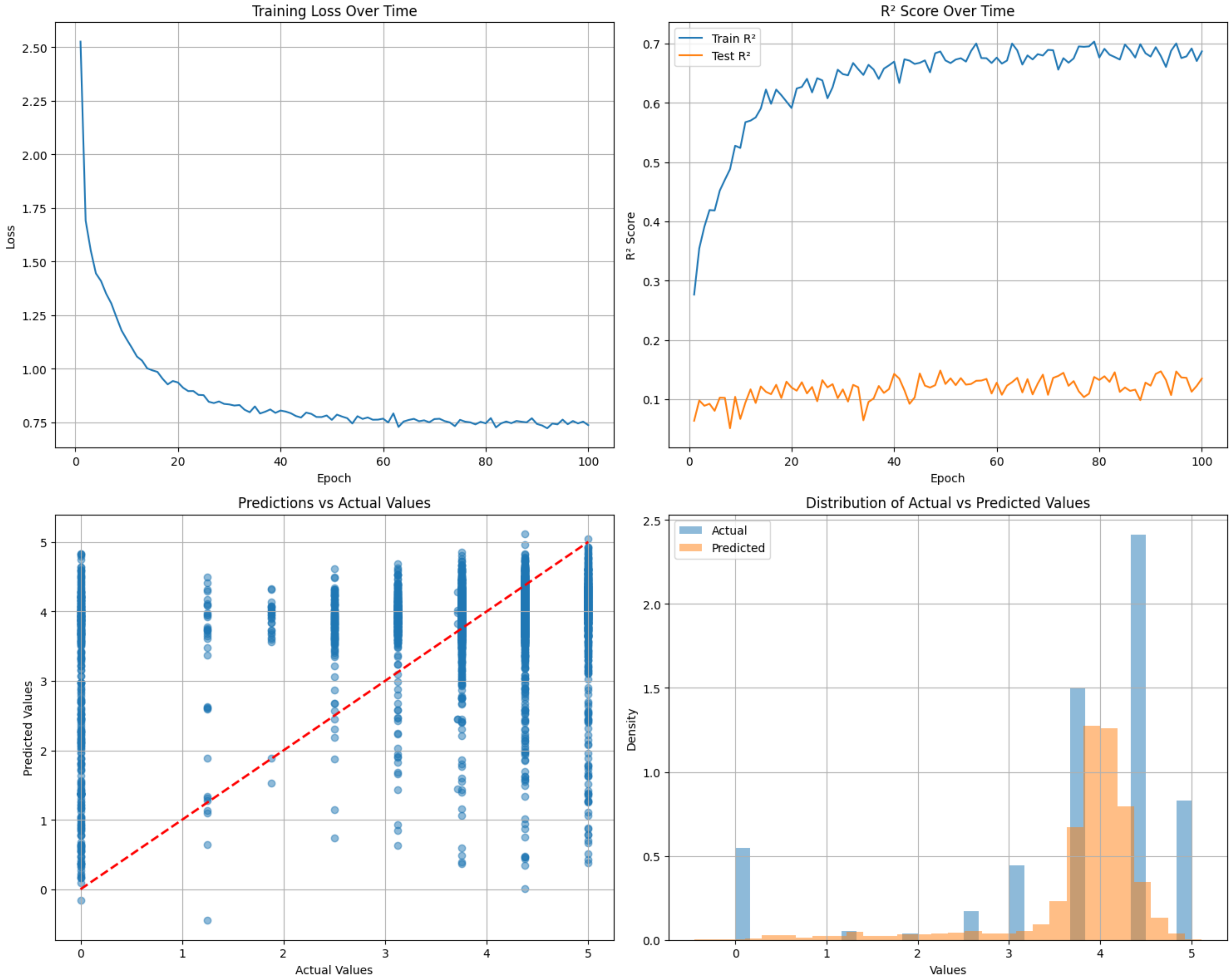
```
model_ML = nn.Sequential(
        nn.Linear(input_size, 128),
        nn.ReLU(),
        nn.Dropout(0.3),  # Add dropout
        nn.Linear(128, 64),
        nn.ReLU(),
        nn.Dropout(0.3),  # Add dropout
        nn.Linear(64, 1),
)
```

	Train R^2	**Test R^2**
TD-IDF	0,6869	0,1346
Word2Vec	0,206	0,1339
BERT	0,2843	0,0985

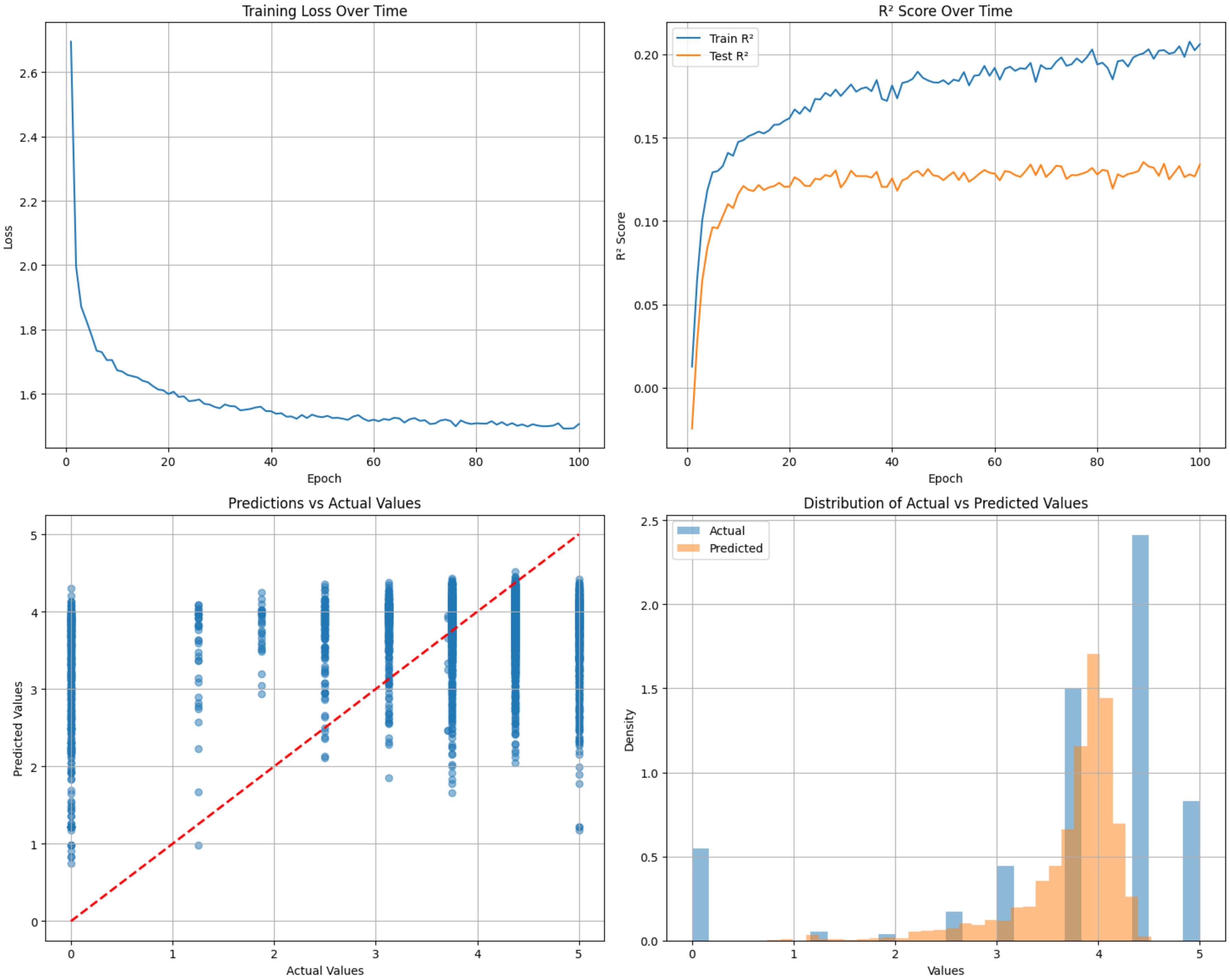
BERT


Training Loss Over Time
Loss
2.4
2.2
2.0
1.8
1.6
1.4
Epoch
0
20
40
60
80
100
R² Score Over Time
Train R²
Test R²
R² Score
0.25
0.20
0.15
0.10
0.05
Epoch
0
20
40
60
80
100
Predictions vs Actual Values
Predicted Values
5
4
3
2
1
0
−1
−2
Actual Values
0
1
2
3
4
5
Distribution of Actual vs Predicted Values
Actual
Predicted
Density
2.5
2.0
1.5
1.0
0.5
0.0
Values
−2
−1
0
1
2
3
4
5

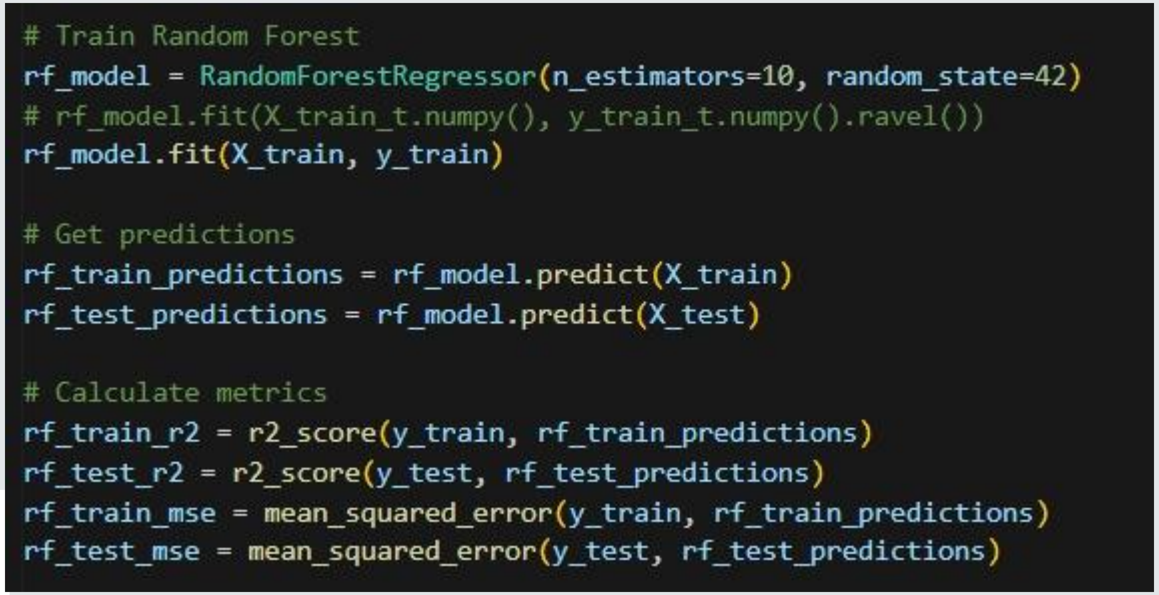
TF-IDF


Training Loss Over Time
Loss
Epoch
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0.75
0
20
40
60
80
100
R² Score Over Time
Train R²
Test R²
R² Score
Epoch
0.7
0.6
0.5
0.4
0.3
0.2
0.1
Predictions vs Actual Values
Predicted Values
Actual Values
5
4
3
2
1
0
Distribution of Actual vs Predicted Values
Actual
Predicted
Density
Values
2.5
2.0
1.5
1.0
0.5
0.0

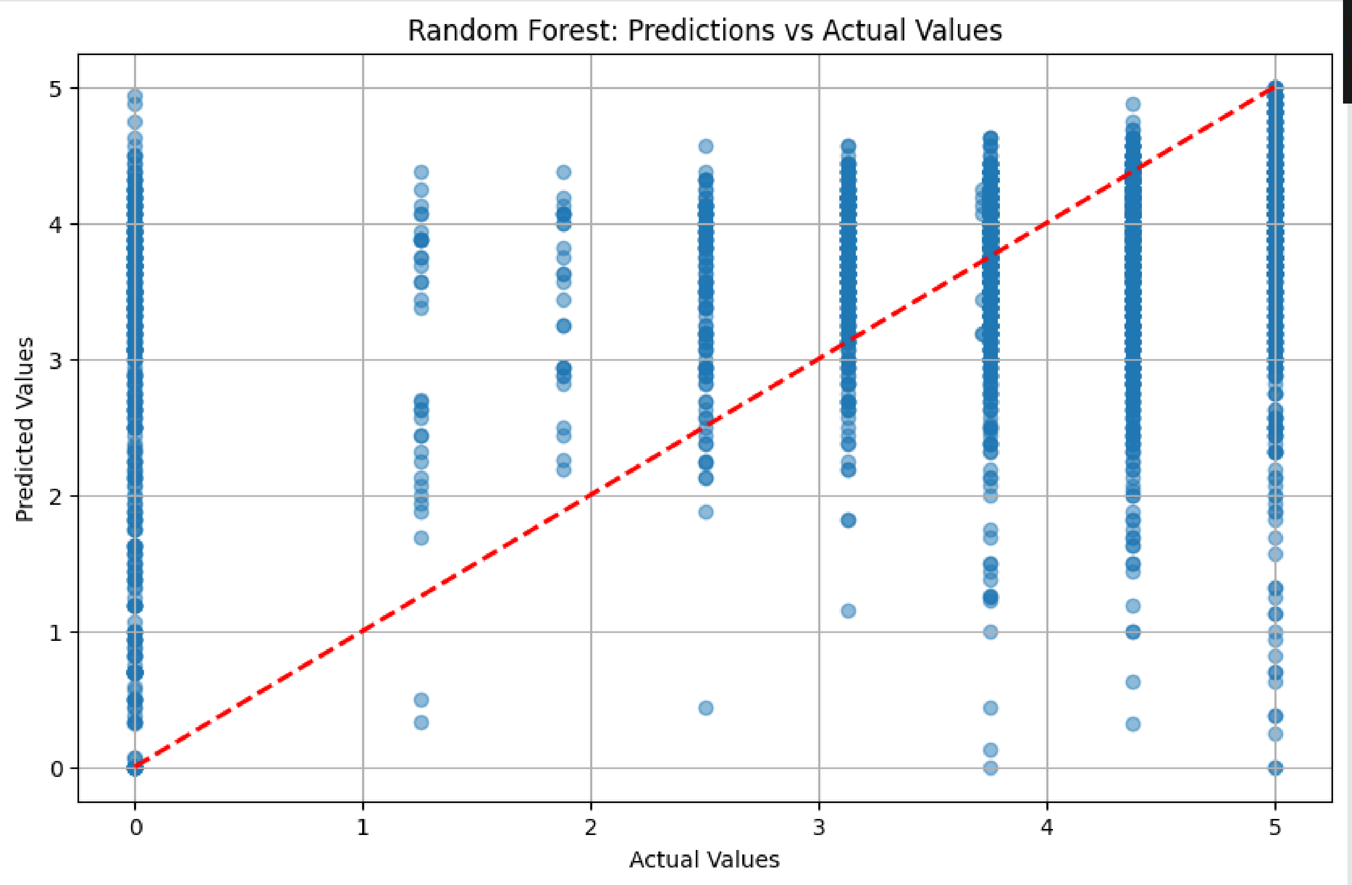
Word2Vec


Training Loss Over Time
Loss
2.6
2.4
2.2
2.0
1.8
1.6
Epoch
0
20
40
60
80
100
R² Score Over Time
Train R²
Test R²
R² Score
0.20
0.15
0.10
0.05
0.00
Epoch
0
20
40
60
80
100
Predictions vs Actual Values
Predicted Values
Actual Values
0
1
2
3
4
5
Distribution of Actual vs Predicted Values
Actual
Predicted
Density
2.5
2.0
1.5
1.0
0.5
0.0
Values
0
1
2
3
4
5

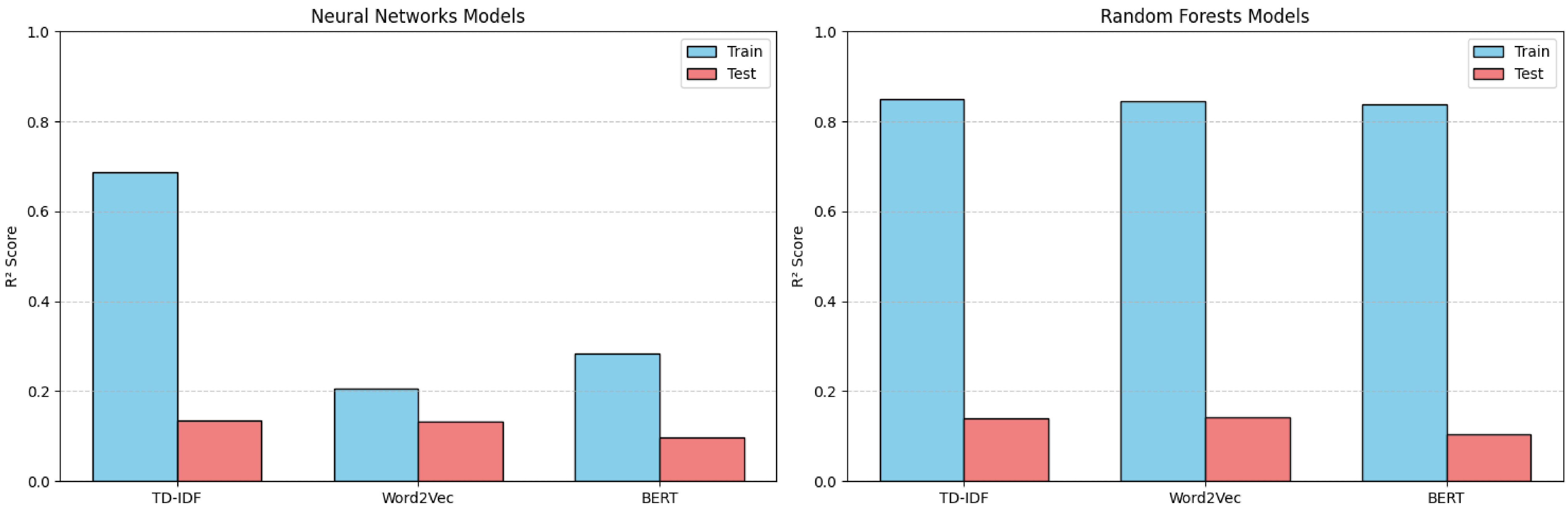
Random Forest



```
# Train Random Forest
rf_model = RandomForestRegressor(n_estimators=10, random_state=42)
# rf_model.fit(X_train_t.numpy(), y_train_t.numpy().ravel())
rf_model.fit(X_train, y_train)

# Get predictions
rf_train_predictions = rf_model.predict(X_train)
rf_test_predictions = rf_model.predict(X_test)

# Calculate metrics
rf_train_r2 = r2_score(y_train, rf_train_predictions)
rf_test_r2 = r2_score(y_test, rf_test_predictions)
rf_train_mse = mean_squared_error(y_train, rf_train_predictions)
rf_test_mse = mean_squared_error(y_test, rf_test_predictions)
```

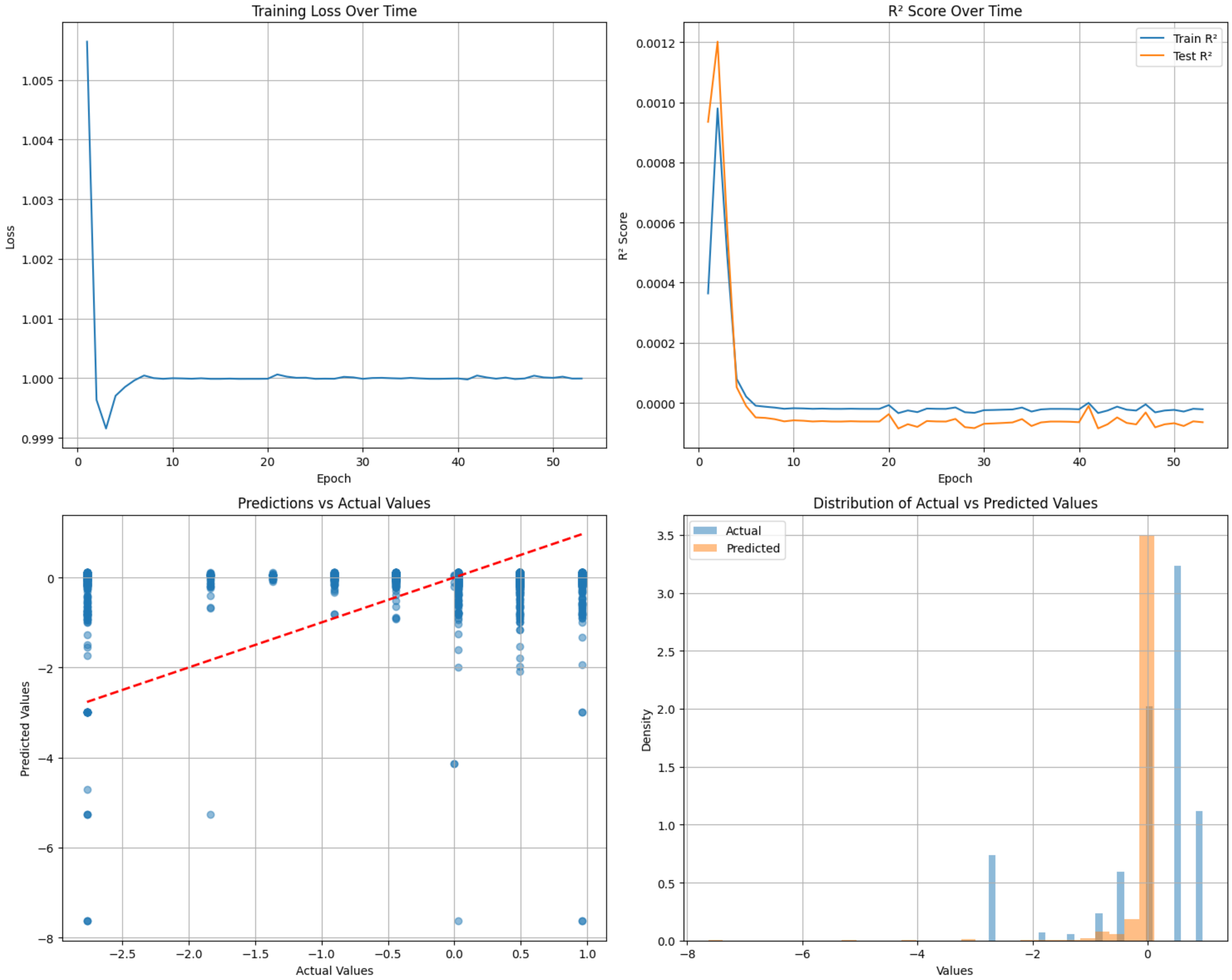


	Train R^2	Test R^2
TD-IDF	0,8489	0,1388
Word2Vec	0,844	0,1419
BERT	0,8378	0,1037


Neural Networks Models
Random Forests Models
R² Score
1.0
0.8
0.6
0.4
0.2
0.0
Train
Test
TD-IDF
Word2Vec
BERT

Hugging Face

We used RoBERTa as a model but the results were not better



Conclusion

Problems

- Bad regression model results.

Possible solutions

- Try different models.
- Find correlated inputs.

THANK YOU VERY MUCH

Iker Aldasoro

Jon Lejardi