Build_Regression_Models

March 18, 2025

1 Analysis of Regression Models for Abalone Age Prediction

[23]:
```
__author__ = "Will Hinton"
__email__ = "willhint@gmail.com"
__website__ = "whinton0.github.io/py"
```

[24]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_log_error
import ace_tools_open as tools
```

1.1 Load datasets into DataFrames

- train.csv contains training data with features and target variables (Rings)
- test.csv contains test data where the target variable needs to be predicted

[25]:
```
train_df = pd.read_csv("train.csv")
test_df = pd.read_csv("test.csv")
sample_submission = pd.read_csv("sample_submission.csv")
```

1.2 Encode categorical variable

- Encode categorical variable 'Sex' using one-hot encoding
- This converts the categorical 'Sex' column into numerical values

[26]:
```
train_df = pd.get_dummies(train_df, columns=['Sex'], drop_first=True)
test_df = pd.get_dummies(test_df, columns=['Sex'], drop_first=True)
```

1.3 Define features and target variable

- 'id' is dropped as it is not a relevant feature

[27]:
```
X = train_df.drop(columns=['id', 'Rings'])  # Feature set
y = train_df['Rings']  # Target variable
```

1.4 Split into training and validation sets

- Split into training and validation sets (80% train, 20% validation)
- This helps evaluate model performance before applying to the test set

[28]:
```
X_train, X_valid, y_train, y_valid = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

2 Standardize numerical features

- Scaling ensures all features contribute equally to the model

[29]:
```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_valid_scaled = scaler.transform(X_valid)
X_test_scaled = scaler.transform(test_df.drop(columns=['id'],␣
↪errors='ignore'))  # Apply same scaling to test set
```

2.1 Define RMSLE

- Define RMSLE function (Root Mean Squared Logarithmic Error)
- RMSLE penalizes under-predictions more than over-predictions
- It helps measure model accuracy for data with skewed distributions

[30]:
```
def rmsle(y_true, y_pred):
    return np.sqrt(mean_squared_log_error(y_true, np.maximum(y_pred, 0)))  #␣
↪Only non-negative predictions
```

2.2 Train Linear Regression Model

- Simple regression model assumes a linear relationship between features and target

[31]:
```
lr_model = LinearRegression()
lr_model.fit(X_train_scaled, y_train)
y_pred_lr = lr_model.predict(X_valid_scaled)
```

2.3 Train Random Forest Model

- An ensemble learning method that builds multiple decision trees to improve predictions (James et al. ISLP, 2023)

[ ]:
```
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train_scaled, y_train)
y_pred_rf = rf_model.predict(X_valid_scaled)
```

Evaluate models using RMSLE

[ ]:
```
rmsle_lr = rmsle(y_valid, y_pred_lr)  # Evaluate Linear Regression
rmsle_rf = rmsle(y_valid, y_pred_rf)  # Evaluate Random Forest
```

2.4 Prepare results for display

- Compare the performance of the two models based on RMSLE
- Display results

[ ]:
```
results_df = pd.DataFrame({
    "Model": ["Linear Regression", "Random Forest"],
    "RMSLE": [rmsle_lr, rmsle_rf]
})

#tools.display_dataframe_to_user(name="Model Evaluation (RMSLE)",␣
 ↪dataframe=results_df)
results_df
```

2.5 Make predictions - Linear Regression

- Make predictions on the test dataset using Linear Regression
- Predict Rings and ensure values are non-negative integers

[ ]:
```
test_df_lr = test_df[['id']].copy()
test_df_lr['Rings'] = np.maximum(lr_model.predict(X_test_scaled), 0).astype(int)
submission_lr_path = "submission_lr.csv"
test_df_lr.to_csv(submission_lr_path, index=False)
```

2.6 Make predictions - Random Forest

- Make predictions on the test dataset using Random Forest
- Predict Rings and ensure values are non-negative integers

[ ]:
```
test_df_rf = test_df[['id']].copy()
test_df_rf['Rings'] = np.maximum(rf_model.predict(X_test_scaled), 0).astype(int)
submission_rf_path = "submission_rf.csv"
test_df_rf.to_csv(submission_rf_path, index=False)
```

2.7 Provide download links for the submission files

[ ]:
```
submission_lr_path, submission_rf_path
```

[ ]:
```
# --- Recommendations to Improve RMSLE ---
# 1. Hyperparameter Tuning:
#    - Use GridSearchCV or RandomizedSearchCV to find the optimal model␣
 ↪parameters.
# 2. Feature Engineering:
#    - Generate new meaningful features, such as ratios between weights.
# 3. Outlier Detection:
#    - Identify and remove or adjust outliers that could be skewing␣
 ↪predictions.
# 4. Log Transformation:
```

```
#    - Apply log transformation to Rings to stabilize variance and improve␣
↪model performance.
# 5. Increase Model Complexity:
#    - Try boosting algorithms like XGBoost or LightGBM for better performance.
# 6. Cross-Validation:
#    - Use k-fold cross-validation to improve generalization and avoid␣
↪overfitting.
# 7. Ensemble Methods:
#    - Combine multiple models (e.g., blend Linear Regression and Random␣
↪Forest) to reduce bias and variance.
```