Untitled

December 5, 2025

[7]:

```python
import pandas as pd
import numpy as np
from sklearn.ensemble import GradientBoostingRegressor, RandomForestRegressor,␣
 ↪VotingRegressor
from sklearn.model_selection import KFold

# 1. Load Data
train_df = pd.read_csv('train.csv')
test_df = pd.read_csv('test.csv')

# ---------------------------------------------------------
# SIMPLE PREPROCESSING (Back to Basics)
# ---------------------------------------------------------
def process_data(df):
    # Fix Horsepower
    df['horsepower'] = pd.to_numeric(df['horsepower'], errors='coerce')
    df['horsepower'] = df['horsepower'].fillna(df['horsepower'].median())

    # Physics: Log Weight is the most important feature
    df['log_weight'] = np.log(df['weight'])

    # Physics: Power-to-Weight
    df['power_to_weight'] = df['horsepower'] / df['weight']

    return df

train_df = process_data(train_df)
test_df = process_data(test_df)

# ---------------------------------------------------------
# FEATURES
# ---------------------------------------------------------
# Trees handle 'origin' and 'cylinders' as numbers just fine.
# We don't need to change them to categories.
features = ['displacement', 'horsepower', 'log_weight', 'acceleration', 'year',␣
 ↪'origin', 'cylinders', 'power_to_weight']
```

```python
X_train_full = train_df[features]
y_train_full = train_df['mpg']
X_test = test_df[features]

# Safety Check: Fill gaps
X_train_full = X_train_full.fillna(X_train_full.median())
X_test = X_test.fillna(X_train_full.median())

# -----------------------------------------------------------
# DEFINE THE STABLE MODELS
# -----------------------------------------------------------
def get_model():
    # Model 1: Gradient Boosting (The Smart One)
    gbr = GradientBoostingRegressor(
        n_estimators=500,
        learning_rate=0.02, # Slow and steady learning prevents "exploding"␣
 ↪scores
        max_depth=3,
        random_state=42
    )

    # Model 2: Random Forest (The Stable One)
    # Random Forest is very hard to confuse. It averages 200 random decision␣
 ↪trees.
    rf = RandomForestRegressor(
        n_estimators=200,
        max_depth=10,
        random_state=42
    )

    return VotingRegressor([('gbr', gbr), ('rf', rf)])

# -----------------------------------------------------------
# 10-FOLD BAGGING (To smooth out errors)
# -----------------------------------------------------------
folds = 10
kf = KFold(n_splits=folds, shuffle=True, random_state=42)
test_predictions_sum = np.zeros(len(X_test))

print(f"Starting {folds}-Fold Training (Recovery Mode)...")

for fold, (train_index, val_index) in enumerate(kf.split(X_train_full,␣
 ↪y_train_full)):

    X_fold = X_train_full.iloc[train_index]
    y_fold = y_train_full.iloc[train_index]
    # Log Transform Target (Still the best trick for this dataset)
    y_fold_log = np.log(y_fold)

    model = get_model()
    model.fit(X_fold, y_fold_log)

    log_pred = model.predict(X_test)
    test_predictions_sum += np.exp(log_pred)

    print(f" - Fold {fold+1} complete.")

# Average
final_predictions = test_predictions_sum / folds

# ---------------------------------------------------------
# SAVE
# ---------------------------------------------------------
submission = pd.DataFrame({
    'ID': test_df['ID'],
    'mpg': final_predictions
})

submission.to_csv('submission.csv', index=False)

print("\n  Success! New 'submission.csv' generated.")
print(submission.head())
```

```
Starting 10-Fold Training (Recovery Mode)…
 - Fold 1 complete.
 - Fold 2 complete.
 - Fold 3 complete.
 - Fold 4 complete.
 - Fold 5 complete.
 - Fold 6 complete.
 - Fold 7 complete.
 - Fold 8 complete.
 - Fold 9 complete.
 - Fold 10 complete.

 Success! New 'submission.csv' generated.
                                        ID        mpg
0  70_chevrolet chevelle malibu_alpha_3505  16.487742
1          71_buick skylark 320_bravo_3697  14.606418
2       70_plymouth satellite_charlie_3421  16.242916
3              68_amc rebel sst_delta_3418  16.164524
4                 70_ford torino_echo_3444  16.651944
```