Foundations of AI
Yunyu Guo
Assignment 2
Due September 23

- For each classifier:
  - Report the precision, recall, and F1 score
    See below
  - Describe what the confusion matrix tells you about the performance (1 or 2 sentences)
    See below
  - Write a sentence or so about possible reasons why this model may or may not have been the right model for the task
    See below
- For each regression (numerical output) model:
  - Report the Mean Squared Error
    See below
  - Describe what the scatter plot tells you about the performance (1 sentence)
    See below
  - Write a sentence or so about possible reasons why this model may or may not have been the right model for the task
    See below
- How long did this assignment take you? (1 sentence)
  3 days
- Whom did you work with, and how? (1 sentence each)
  - Discussing the assignment with others is encouraged, as long as you don't share the code.

  I discussed how to upload image data to google colab (using processed arrays) with other student
- Which resources did you use? (1 sentence each)
  - For each, please list the URL and a brief description of how it was useful.

  Dataset: numerical: bone age:
  https://www.kaggle.com/datasets/kmader/rsna-bone-age/data?select=boneage-training-dataset

Classification: dog breeds: https://www.kaggle.com/datasets/khushikhushikhushi/dog-breed-image-dataset
K fold: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html
Train test split: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
GridSearchCV: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

- A few sentences about:
  - What was the most difficult part of the assignment?
    Tuning the hyper-parameters of an estimator, what parameters and value should be used with respect to each model. How to handle long run time, what parameters, number of iterations need to be adjusted.
  - What was the most rewarding part of the assignment?
    To be able to practice using 5 different models for classification and numerical data
  - What did you learn doing the assignment?
    How to deal with excessive memory usage and long run time, to adjust the parameters in the grid search, utilize the preprocessed data and label arrays and store them, so that I don't need to process data every time when test on different models.
  - Constructive and actionable suggestions for improving assignments, office hours, and class time are always welcome.
    Upon searching for how to use k fold cross validation in different models, I find most resources use grid search with different parameter with respect to different models, I feel like this part was not mentioned at class and assignment, it would be better to have a clear instruction as what resources we should look at when turning models.

Classification
I use dog breeds dataset, the dataset contains 10 folders, each folder is named by the dog breed. Total 967 images in the dataset.

Logistic regression:
Error: the filenames in your dataset contain breed names, which cannot be converted to integers. This is causing the int(filename.split('_')[0]) line to fail.
How to fix: Create a mapping from breed names to integers.

Error: images list contains elements of different shapes, which prevents it from being converted to a NumPy array.

How to fix: probably because the images are not of the same size. So I resize all images to the same dimensions, and flatten the resized images.

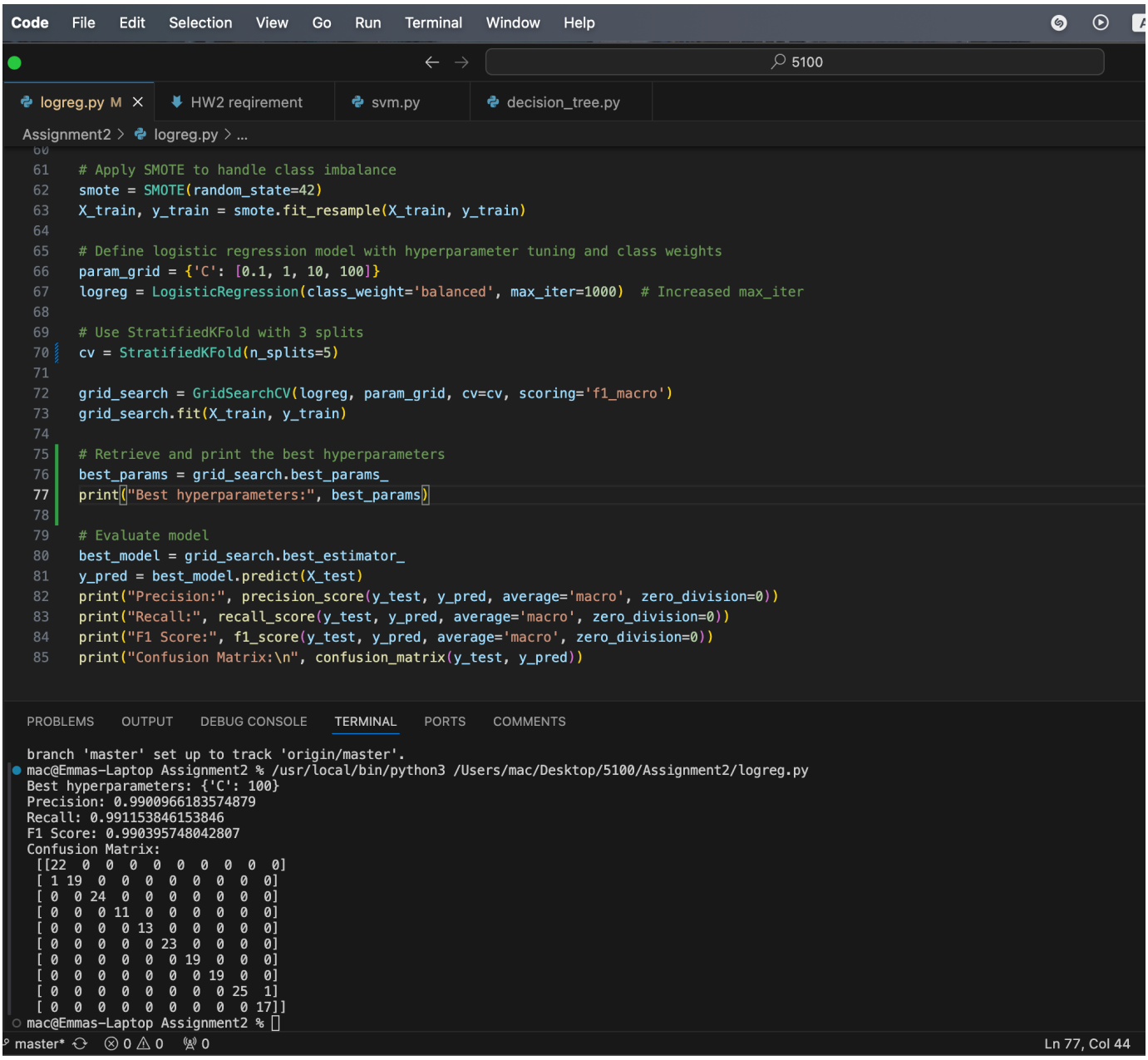
Error: The least populated class in y has only 1 members, which is less than n_splits=2.(I changed n_splits to 3 and 5 later, and the model seemed to work well.) Precision: 0.0 Recall: 0.0 F1 Score: 0.0 Confusion Matrix: [[0 0 1 0 0 0 0] [0 0 1 0 0 0 0] [0 0 0 0 0 0 0] [0 0 0 0 0 0 0] [0 0 0 0 0 0 0] [0 0 0 1 0 0 0] [0 0 0 0 1 0 0]]

How to fix: Added SMOTE to oversample the minority classes in the training set.
Use Class Weights: Set class_weight='balanced' in the LogisticRegression model to handle class imbalance.
Handle Undefined Metrics: Use zero_division=0 in the scoring functions to handle cases where precision or recall is undefined.

Output: when k =3 in k-fold cross validation (first output), it seems precision, recall, and F1 scores are very high, the model is performing well. The confusion matrix also shows that most of the predictions are correct, with very few misclassifications.

Best hyperparameters: {'C': 100}. During K-fold cross-validation, different values of C are tested to find the one that yields the best performance on the validation sets. The cross-validation process indicated that this value of C resulted in the best performance in terms of the F1 macro score, the model generalizes well to unseen data with this level of regularization.

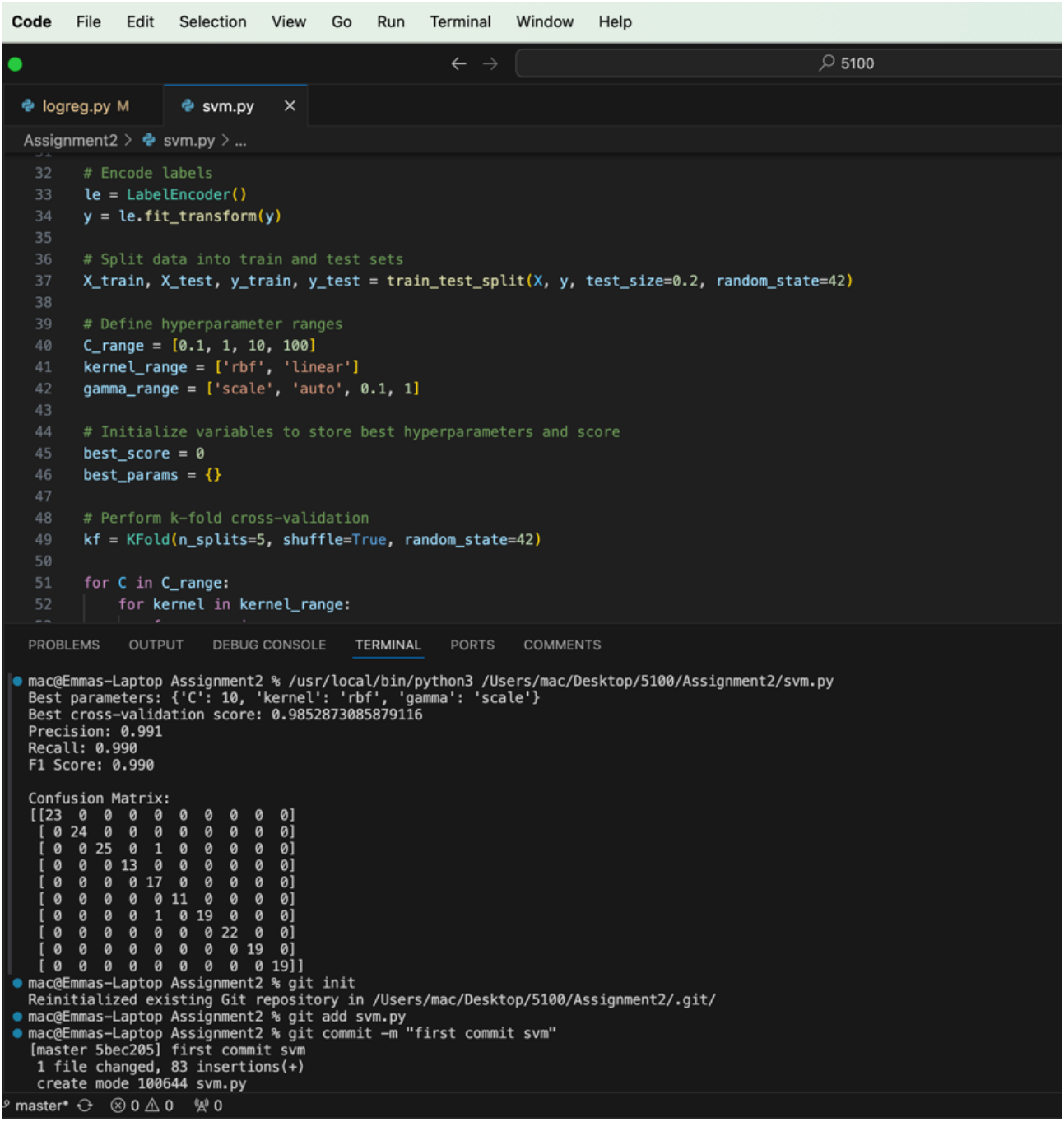
When k=5 (second output), the confusion matrix indicates:
Class 0: 22 correctly classified, 0 misclassified.
Class 1: 19 correctly classified, 1 misclassified as class 0.
Class 2: 24 correctly classified, 0 misclassified.
Class 3: 11 correctly classified, 0 misclassified.
Class 4: 13 correctly classified, 0 misclassified.
Class 5: 23 correctly classified, 0 misclassified.
Class 6: 19 correctly classified, 0 misclassified.
Class 7: 19 correctly classified, 0 misclassified.
Class 8: 25 correctly classified, 1 misclassified as class 9.
Using 5-fold cross-validation provides a more robust estimate of the model's performance, reducing the likelihood of overfitting.


Code File Edit Selection View Go Run Terminal Window Help
5100
logreg.py M
HW2 reqirement
svm.py
decision_tree.py
Assignment2 > logreg.py > ...
# Apply SMOTE to handle class imbalance
smote = SMOTE(random_state=42)
X_train, y_train = smote.fit_resample(X_train, y_train)
# Define logistic regression model with hyperparameter tuning and class weights
param_grid = {'C': [0.1, 1, 10, 100]}
logreg = LogisticRegression(class_weight='balanced', max_iter=1000) # Increased max_iter
# Use StratifiedKFold with 3 splits
cv = StratifiedKFold(n_splits=5)
grid_search = GridSearchCV(logreg, param_grid, cv=cv, scoring='f1_macro')
grid_search.fit(X_train, y_train)
# Retrieve and print the best hyperparameters
best_params = grid_search.best_params_
print("Best hyperparameters:", best_params)
# Evaluate model
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)
print("Precision:", precision_score(y_test, y_pred, average='macro', zero_division=0))
print("Recall:", recall_score(y_test, y_pred, average='macro', zero_division=0))
print("F1 Score:", f1_score(y_test, y_pred, average='macro', zero_division=0))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
branch 'master' set up to track 'origin/master'.
mac@Emmas-Laptop Assignment2 % /usr/local/bin/python3 /Users/mac/Desktop/5100/Assignment2/logreg.py
Best hyperparameters: {'C': 100}
Precision: 0.9900966183574879
Recall: 0.991153846153846
F1 Score: 0.990395748042807
Confusion Matrix:
[[22 0 0 0 0 0 0 0 0 0]
[1 19 0 0 0 0 0 0 0 0]
[0 0 24 0 0 0 0 0 0 0]
[0 0 0 11 0 0 0 0 0 0]
[0 0 0 0 13 0 0 0 0 0]
[0 0 0 0 0 23 0 0 0 0]
[0 0 0 0 0 0 19 0 0 0]
[0 0 0 0 0 0 0 19 0 0]
[0 0 0 0 0 0 0 0 25 1]
[0 0 0 0 0 0 0 0 0 17]]
mac@Emmas-Laptop Assignment2 %
master* 0 0 0
Ln 77, Col 44

Support vector machine:
Best parameters: {'C': 10, 'kernel': 'rbf', 'gamma': 'scale'}
C: The regularization parameter. A value of 10 indicates a moderate level of regularization, balancing the trade-off between achieving a low training error and a low testing error.
kernel: The kernel type used in the algorithm. 'rbf' (Radial Basis Function) is a popular choice for non-linear data.
gamma: The kernel coefficient. 'scale' means this value adjusts the influence of each training example based on the variance of the input features, leading to a balanced decision boundary.

Confusion matrix:
Class 0: 23 correctly classified, 0 misclassified.
Class 1: 24 correctly classified, 0 misclassified.
Class 2: 25 correctly classified, 1 misclassified as class 4.
Class 3: 13 correctly classified, 0 misclassified.
Class 4: 17 correctly classified, 0 misclassified.
Class 5: 11 correctly classified, 0 misclassified.
Class 6: 19 correctly classified, 1 misclassified as class 4.
Class 7: 22 correctly classified, 0 misclassified.
Class 8: 19 correctly classified, 0 misclassified.
Class 9: 19 correctly classified, 0 misclassified.

The SVM model is performing well with high precision, recall, and F1 scores.
The confusion matrix shows very few misclassifications, the model is accurately predicting the classes.
Using 5-fold cross-validation provides a more robust estimate of the model's performance, reducing the likelihood of overfitting.

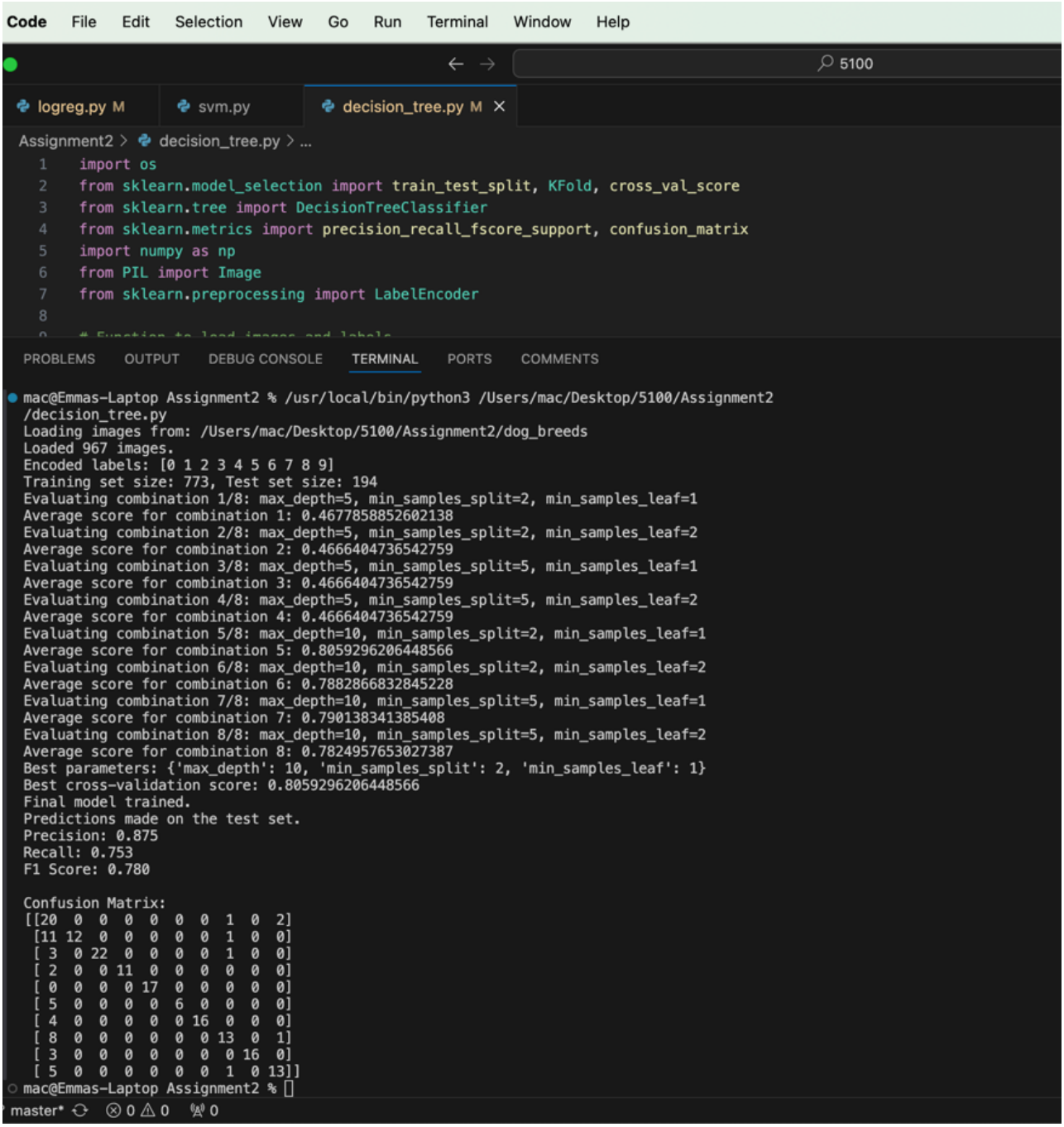

Code File Edit Selection View Go Run Terminal Window Help
5100
logreg.py M
svm.py
Assignment2 > svm.py > ...
# Encode labels
le = LabelEncoder()
y = le.fit_transform(y)
# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Define hyperparameter ranges
C_range = [0.1, 1, 10, 100]
kernel_range = ['rbf', 'linear']
gamma_range = ['scale', 'auto', 0.1, 1]
# Initialize variables to store best hyperparameters and score
best_score = 0
best_params = {}
# Perform k-fold cross-validation
kf = KFold(n_splits=5, shuffle=True, random_state=42)
for C in C_range:
for kernel in kernel_range:
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
mac@Emmas-Laptop Assignment2 % /usr/local/bin/python3 /Users/mac/Desktop/5100/Assignment2/svm.py
Best parameters: {'C': 10, 'kernel': 'rbf', 'gamma': 'scale'}
Best cross-validation score: 0.9852873085879116
Precision: 0.991
Recall: 0.990
F1 Score: 0.990
Confusion Matrix:
[[23 0 0 0 0 0 0 0 0 0]
[0 24 0 0 0 0 0 0 0 0]
[0 0 25 0 1 0 0 0 0 0]
[0 0 0 13 0 0 0 0 0 0]
[0 0 0 0 17 0 0 0 0 0]
[0 0 0 0 0 11 0 0 0 0]
[0 0 0 0 1 0 19 0 0 0]
[0 0 0 0 0 0 0 22 0 0]
[0 0 0 0 0 0 0 0 19 0]
[0 0 0 0 0 0 0 0 0 19]]
mac@Emmas-Laptop Assignment2 % git init
Reinitialized existing Git repository in /Users/mac/Desktop/5100/Assignment2/.git/
mac@Emmas-Laptop Assignment2 % git add svm.py
mac@Emmas-Laptop Assignment2 % git commit -m "first commit svm"
[master 5bec205] first commit svm
1 file changed, 83 insertions(+)
create mode 100644 svm.py
master* 0 0

Decision tree:
Issue: first version, the code keeps running and nothing print out, this is the first version of tree:
max_depth_range = [5, 10, 15, 20, None]
min_samples_split_range = [2, 5, 10]
min_samples_leaf_range = [1, 2, 4]

debug the tree:
max_depth_range = [5, 10] # Reduced for debugging
min_samples_split_range = [2, 5] # Reduced for debugging
min_samples_leaf_range = [1, 2] # Reduced for debugging

It seems decision tree model takes more time to run comparing to Logistic and SVM models.
Precision: A precision of 0.875 means that 87.5% of the instances predicted as positive are actually positive.
Recall: A recall of 0.753 means that 75.3% of the actual positive instances are correctly identified by the model.
F1 score: F1 Score of 0.780 indicates a balance between precision and recall, the model performs reasonably well in both identifying positive instances and minimizing false positives.

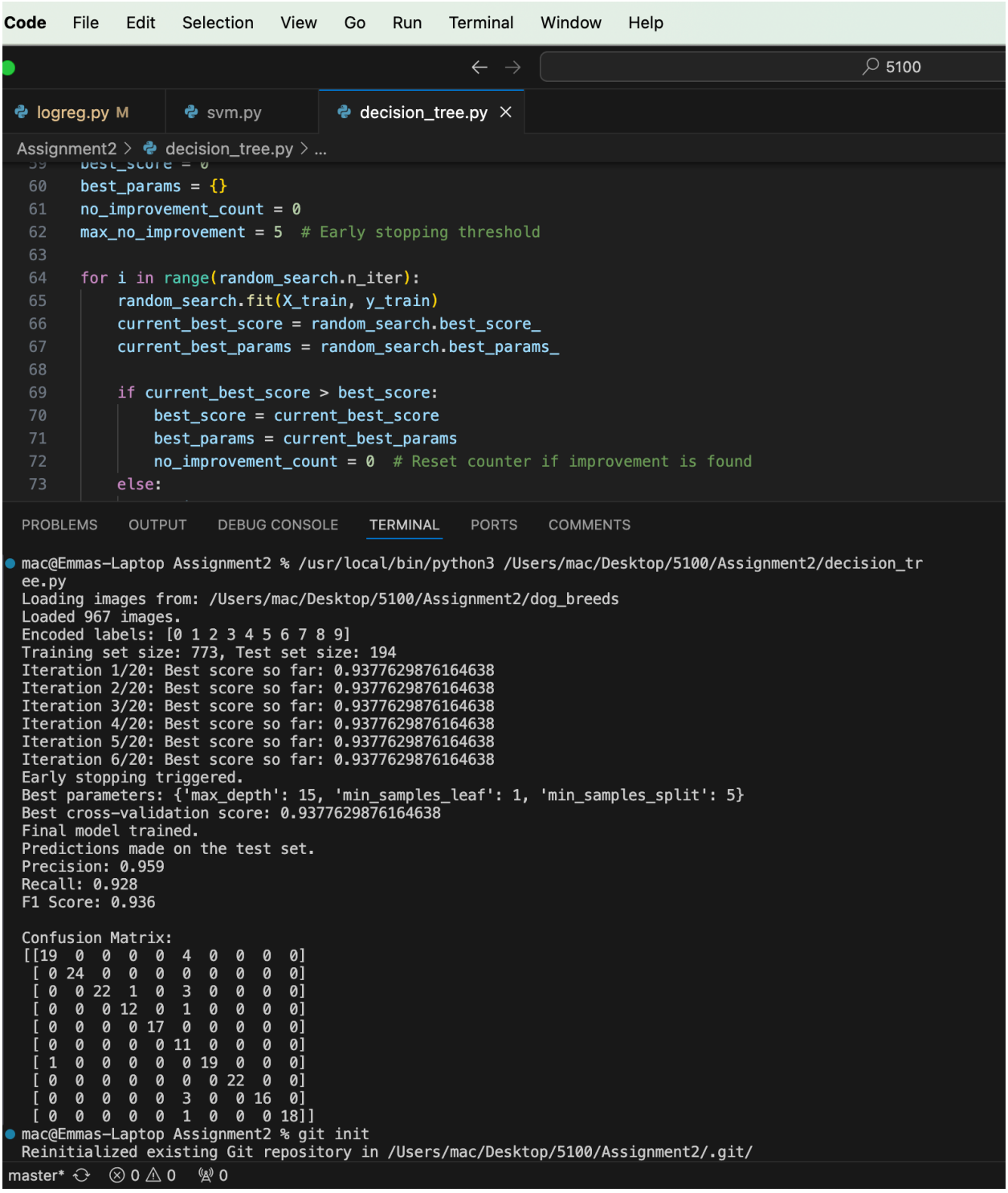
Confusion matrix: misclassified classes 1 and 7
Class 0: 20 correctly classified, 3 misclassified.
Class 1: 12 correctly classified, 12 misclassified.
Class 2: 22 correctly classified, 4 misclassified.
Class 3: 11 correctly classified, 2 misclassified.
Class 4: 17 correctly classified, 0 misclassified.
Class 5: 6 correctly classified, 5 misclassified.
Class 6: 16 correctly classified, 4 misclassified.
Class 7: 13 correctly classified, 9 misclassified.
Class 8: 16 correctly classified, 3 misclassified.
Class 9: 13 correctly classified, 6 misclassified.



After debugging, optimize decision tree model by the followings: Parallel Processing: within the RandomizedSearchCV by setting the n_jobs parameter to -1. Use all available CPU cores to perform the search.
Randomized Search: Use RandomizedSearchCV instead of GridSearchCV to sample a subset of the hyperparameter space.
Early Stopping: Implement early stopping to terminate evaluations that are unlikely to have good results.

Output: The best score did not improve over six iterations, triggering early stopping.
Best parameters: {'max_depth': 15, 'min_samples_leaf': 1, 'min_samples_split': 5}
The best hyperparameters found were max_depth=15, min_samples_leaf=1, and min_samples_split=5.
Best cross-validation score: 0.9377629876164638
The highest average cross-validation score achieved was approximately 0.938.
Confusion matrix:
Class 0: 19 correctly classified, 4 misclassified.
Class 1: 24 correctly classified, 0 misclassified.
Class 2: 22 correctly classified, 4 misclassified.
Class 3: 12 correctly classified, 1 misclassified.
Class 4: 17 correctly classified, 0 misclassified.
Class 5: 11 correctly classified, 0 misclassified.
Class 6: 19 correctly classified, 1 misclassified.
Class 7: 22 correctly classified, 0 misclassified.
Class 8: 16 correctly classified, 3 misclassified.
Class 9: 18 correctly classified, 1 misclassified.

The model has a high precision of 0.959, most of the positive predictions are correct.
The recall is 0.928, the model is able to identify a large portion of the actual positive instances.
Balanced F1 Score: The F1 Score of 0.936 is a good balance between precision and recall.
The confusion matrix shows that while the model performs well for most classes, it misclassifies classes 0 and 2.


Code File Edit Selection View Go Run Terminal Window Help
5100
logreg.py M
svm.py
decision_tree.py
Assignment2 > decision_tree.py > ...
best_score = 0
best_params = {}
no_improvement_count = 0
max_no_improvement = 5 # Early stopping threshold
for i in range(random_search.n_iter):
random_search.fit(X_train, y_train)
current_best_score = random_search.best_score_
current_best_params = random_search.best_params_
if current_best_score > best_score:
best_score = current_best_score
best_params = current_best_params
no_improvement_count = 0 # Reset counter if improvement is found
else:
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
mac@Emmas-Laptop Assignment2 % /usr/local/bin/python3 /Users/mac/Desktop/5100/Assignment2/decision_tree.py
Loading images from: /Users/mac/Desktop/5100/Assignment2/dog_breeds
Loaded 967 images.
Encoded labels: [0 1 2 3 4 5 6 7 8 9]
Training set size: 773, Test set size: 194
Iteration 1/20: Best score so far: 0.9377629876164638
Iteration 2/20: Best score so far: 0.9377629876164638
Iteration 3/20: Best score so far: 0.9377629876164638
Iteration 4/20: Best score so far: 0.9377629876164638
Iteration 5/20: Best score so far: 0.9377629876164638
Iteration 6/20: Best score so far: 0.9377629876164638
Early stopping triggered.
Best parameters: {'max_depth': 15, 'min_samples_leaf': 1, 'min_samples_split': 5}
Best cross-validation score: 0.9377629876164638
Final model trained.
Predictions made on the test set.
Precision: 0.959
Recall: 0.928
F1 Score: 0.936
Confusion Matrix:
[[19 0 0 0 0 4 0 0 0 0]
[0 24 0 0 0 0 0 0 0 0]
[0 0 22 1 0 3 0 0 0 0]
[0 0 0 12 0 1 0 0 0 0]
[0 0 0 0 17 0 0 0 0 0]
[0 0 0 0 0 11 0 0 0 0]
[1 0 0 0 0 0 19 0 0 0]
[0 0 0 0 0 0 0 22 0 0]
[0 0 0 0 0 3 0 0 16 0]
[0 0 0 0 0 1 0 0 0 18]]
mac@Emmas-Laptop Assignment2 % git init
Reinitialized existing Git repository in /Users/mac/Desktop/5100/Assignment2/.git/
master* 0 0 0

Multi-layer perceptron:

Initially, the script is getting stuck during the GridSearchCV step:

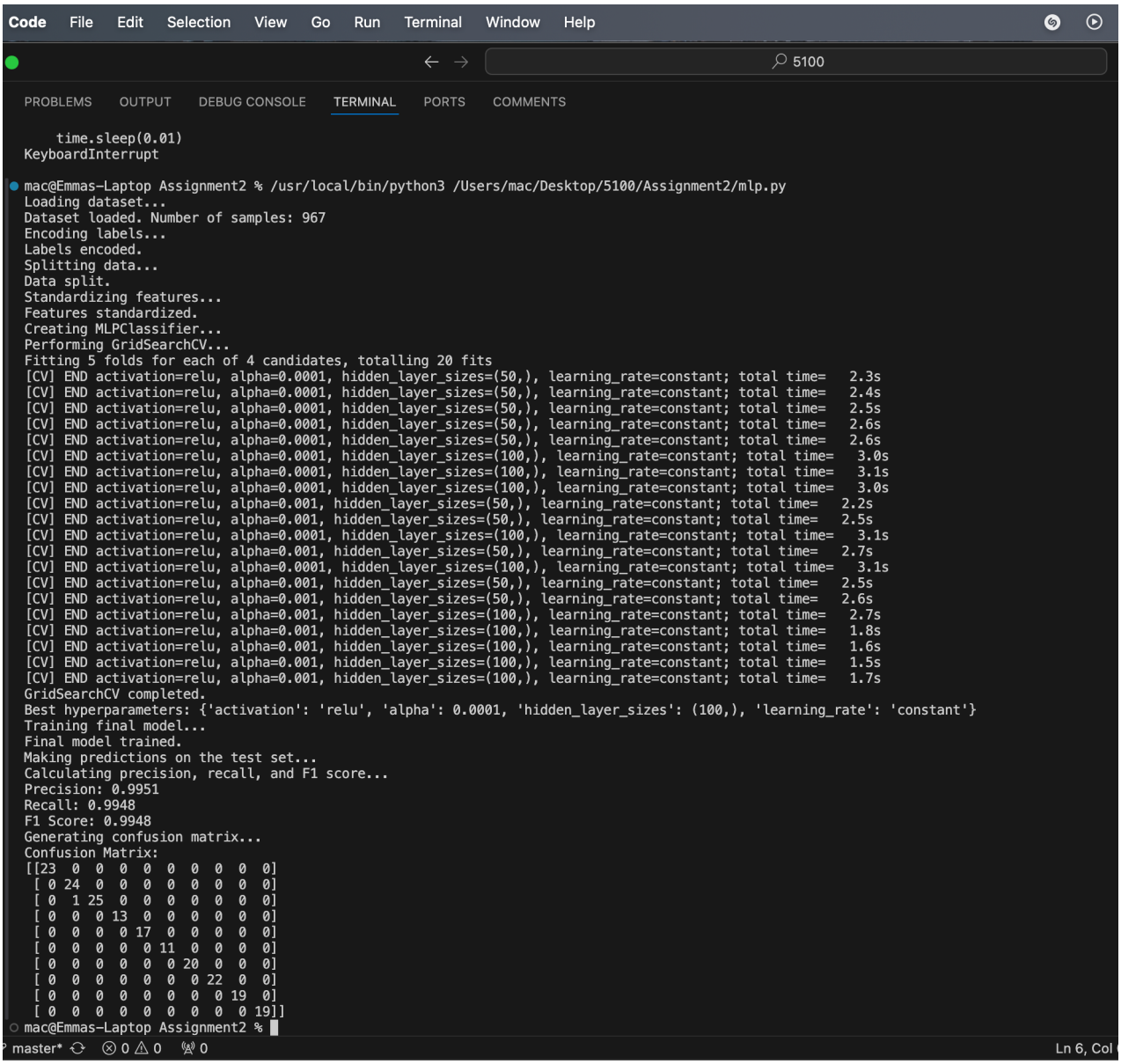
```
# Perform GridSearchCV with 5-fold cross-validation
print("Performing GridSearchCV...")
grid_search = GridSearchCV(mlp, param_grid, cv=5, scoring='f1_weighted', n_jobs=-1)
grid_search.fit(X_train_scaled, y_train)
print("GridSearchCV completed.")
```

how to fix: First, try reducing the parameter grid to speed up the search:

```
param_grid = {
    'hidden_layer_sizes': [(50,), (100,)],
    'activation': ['relu'],
    'alpha': [0.0001, 0.001],
    'learning_rate': ['constant'],
}
```

Add the verbose parameter to GridSearchCV to get more information about its progress:
grid_search = GridSearchCV(mlp, param_grid, cv=5, scoring='f1_weighted', n_jobs=-1, verbose=2).

The debug looks working well, output is in the following print screen.



expanding the grid:

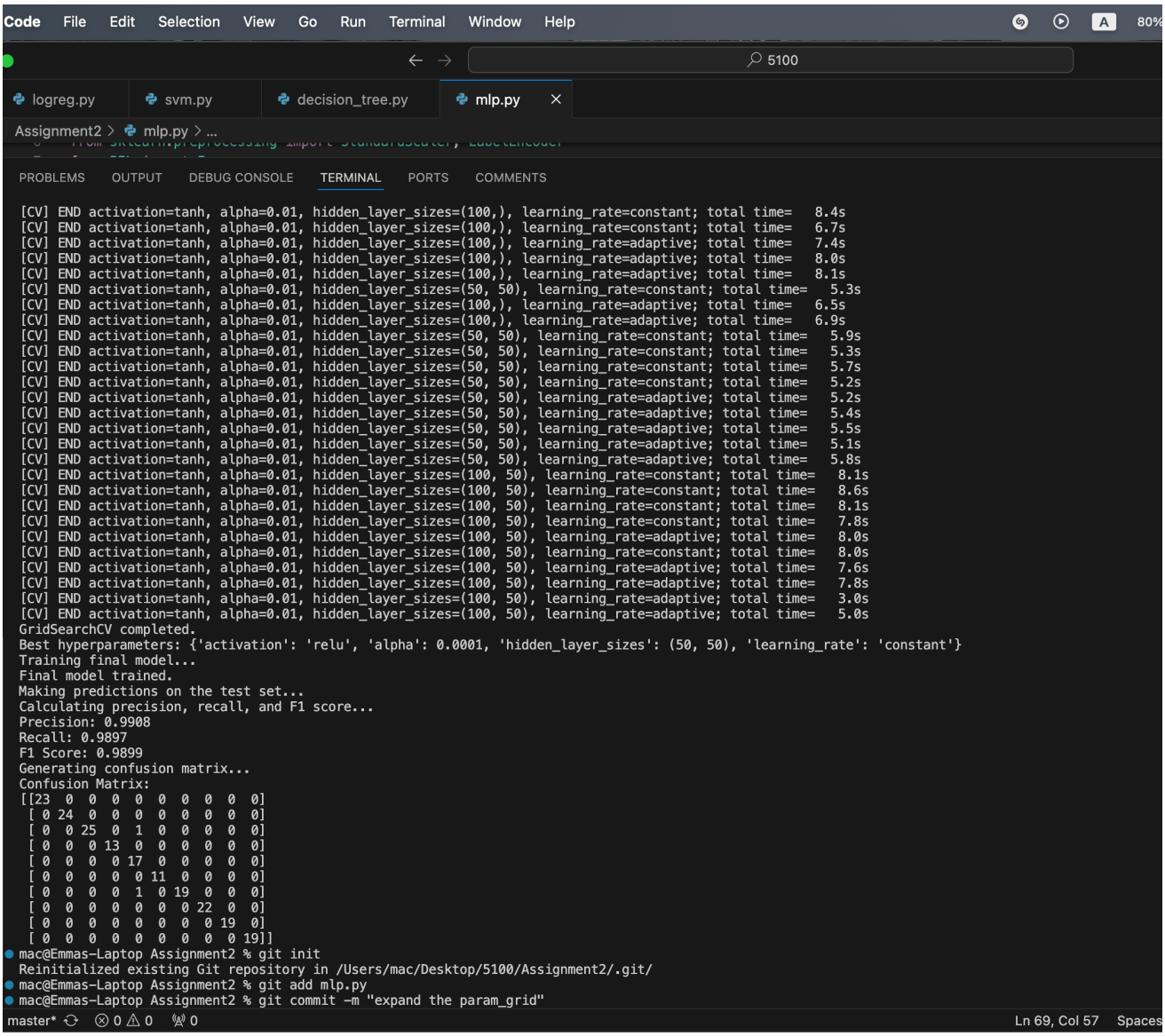
```
param_grid = {
    'hidden_layer_sizes': [(50,), (100,), (50, 50), (100, 50)],
    'activation': ['relu', 'tanh'],
    'alpha': [0.0001, 0.001, 0.01],
    'learning_rate': ['constant', 'adaptive'],
}
```

Output:

Best hyperparameters: {'activation': 'relu', 'alpha': 0.0001, 'hidden_layer_sizes': (50, 50), 'learning_rate': 'constant'}

activation: 'relu': The Rectified Linear Unit (ReLU) activation function was chosen for the hidden layers.
alpha: 0.0001: The regularization term (L2 penalty) is set to 0.0001, which helps prevent overfitting.
hidden_layer_sizes: (50, 50): The neural network has two hidden layers, each with 50 neurons.
learning_rate: 'constant': The learning rate remains constant during training.

Confusion matrix:
Class 0: 23 correct predictions, 0 misclassifications.
Class 1: 24 correct predictions, 0 misclassifications.
Class 2: 25 correct predictions, 1 misclassification (predicted as Class 4).
Class 3: 13 correct predictions, 0 misclassifications.
Class 4: 17 correct predictions, 0 misclassifications.
Class 5: 11 correct predictions, 0 misclassifications.
Class 6: 19 correct predictions, 1 misclassification (predicted as Class 4).
Class 7: 22 correct predictions, 0 misclassifications.
Class 8: 19 correct predictions, 0 misclassifications.
Class 9: 19 correct predictions, 0 misclassifications.


Code File Edit Selection View Go Run Terminal Window Help
80%
5100
logreg.py
svm.py
decision_tree.py
mlp.py
Assignment2 > mlp.py > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
[CV] END activation=tanh, alpha=0.01, hidden_layer_sizes=(100,), learning_rate=constant; total time= 8.4s
[CV] END activation=tanh, alpha=0.01, hidden_layer_sizes=(100,), learning_rate=constant; total time= 6.7s
[CV] END activation=tanh, alpha=0.01, hidden_layer_sizes=(100,), learning_rate=adaptive; total time= 7.4s
[CV] END activation=tanh, alpha=0.01, hidden_layer_sizes=(100,), learning_rate=adaptive; total time= 8.0s
[CV] END activation=tanh, alpha=0.01, hidden_layer_sizes=(100,), learning_rate=adaptive; total time= 8.1s
[CV] END activation=tanh, alpha=0.01, hidden_layer_sizes=(50, 50), learning_rate=constant; total time= 5.3s
[CV] END activation=tanh, alpha=0.01, hidden_layer_sizes=(100,), learning_rate=adaptive; total time= 6.5s
[CV] END activation=tanh, alpha=0.01, hidden_layer_sizes=(100,), learning_rate=adaptive; total time= 6.9s
[CV] END activation=tanh, alpha=0.01, hidden_layer_sizes=(50, 50), learning_rate=constant; total time= 5.9s
[CV] END activation=tanh, alpha=0.01, hidden_layer_sizes=(50, 50), learning_rate=constant; total time= 5.3s
[CV] END activation=tanh, alpha=0.01, hidden_layer_sizes=(50, 50), learning_rate=constant; total time= 5.7s
[CV] END activation=tanh, alpha=0.01, hidden_layer_sizes=(50, 50), learning_rate=constant; total time= 5.2s
[CV] END activation=tanh, alpha=0.01, hidden_layer_sizes=(50, 50), learning_rate=adaptive; total time= 5.2s
[CV] END activation=tanh, alpha=0.01, hidden_layer_sizes=(50, 50), learning_rate=adaptive; total time= 5.4s
[CV] END activation=tanh, alpha=0.01, hidden_layer_sizes=(50, 50), learning_rate=adaptive; total time= 5.5s
[CV] END activation=tanh, alpha=0.01, hidden_layer_sizes=(50, 50), learning_rate=adaptive; total time= 5.1s
[CV] END activation=tanh, alpha=0.01, hidden_layer_sizes=(50, 50), learning_rate=adaptive; total time= 5.8s
[CV] END activation=tanh, alpha=0.01, hidden_layer_sizes=(100, 50), learning_rate=constant; total time= 8.1s
[CV] END activation=tanh, alpha=0.01, hidden_layer_sizes=(100, 50), learning_rate=constant; total time= 8.6s
[CV] END activation=tanh, alpha=0.01, hidden_layer_sizes=(100, 50), learning_rate=constant; total time= 8.1s
[CV] END activation=tanh, alpha=0.01, hidden_layer_sizes=(100, 50), learning_rate=constant; total time= 7.8s
[CV] END activation=tanh, alpha=0.01, hidden_layer_sizes=(100, 50), learning_rate=adaptive; total time= 8.0s
[CV] END activation=tanh, alpha=0.01, hidden_layer_sizes=(100, 50), learning_rate=constant; total time= 8.0s
[CV] END activation=tanh, alpha=0.01, hidden_layer_sizes=(100, 50), learning_rate=adaptive; total time= 7.6s
[CV] END activation=tanh, alpha=0.01, hidden_layer_sizes=(100, 50), learning_rate=adaptive; total time= 7.8s
[CV] END activation=tanh, alpha=0.01, hidden_layer_sizes=(100, 50), learning_rate=adaptive; total time= 3.0s
[CV] END activation=tanh, alpha=0.01, hidden_layer_sizes=(100, 50), learning_rate=adaptive; total time= 5.0s
GridSearchCV completed.
Best hyperparameters: {'activation': 'relu', 'alpha': 0.0001, 'hidden_layer_sizes': (50, 50), 'learning_rate': 'constant'}
Training final model...
Final model trained.
Making predictions on the test set...
Calculating precision, recall, and F1 score...
Precision: 0.9908
Recall: 0.9897
F1 Score: 0.9899
Generating confusion matrix...
Confusion Matrix:
[[23 0 0 0 0 0 0 0 0 0]
[0 24 0 0 0 0 0 0 0 0]
[0 0 25 0 1 0 0 0 0 0]
[0 0 0 13 0 0 0 0 0 0]
[0 0 0 0 17 0 0 0 0 0]
[0 0 0 0 0 11 0 0 0 0]
[0 0 0 0 1 0 19 0 0 0]
[0 0 0 0 0 0 0 22 0 0]
[0 0 0 0 0 0 0 0 19 0]
[0 0 0 0 0 0 0 0 0 19]]
mac@Emmas-Laptop Assignment2 % git init
Reinitialized existing Git repository in /Users/mac/Desktop/5100/Assignment2/.git/
mac@Emmas-Laptop Assignment2 % git add mlp.py
mac@Emmas-Laptop Assignment2 % git commit -m "expand the param_grid"
master* ⊗ 0 ⚠ 0 0
Ln 69, Col 57 Spaces

Random forest:
Initially, the script takes quite a lot of time to run due to extensive 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]
}
```

How to fix: reducing the parameter grid to speed up the search:

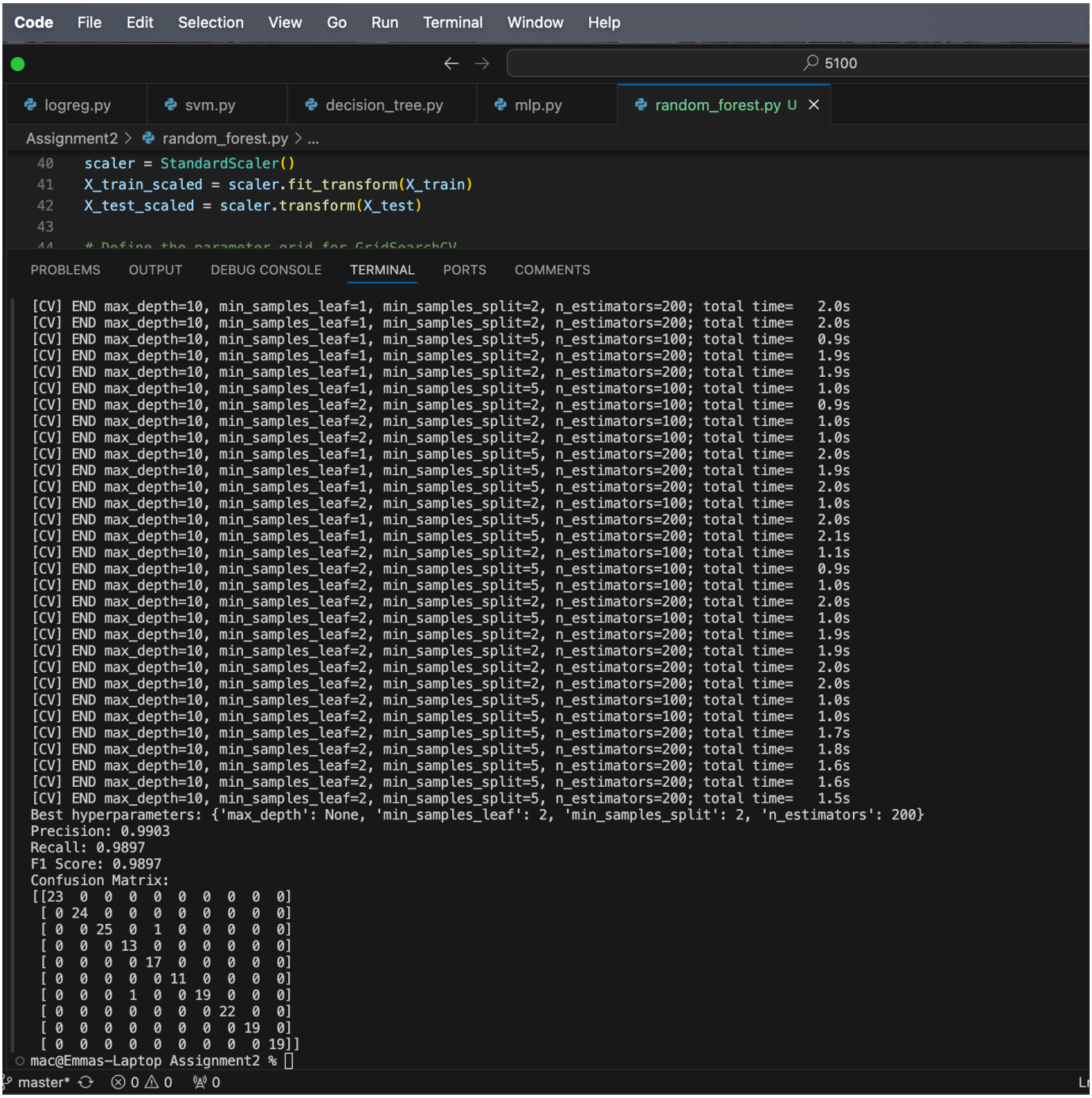
```
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [None, 10],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2]
}
```

Add the verbose parameter to GridSearchCV to get more information about its progress:
grid_search = GridSearchCV(estimator=RandomForestClassifier(), param_grid=param_grid, cv=5, scoring='accuracy', n_jobs=-1, verbose=2)

Output:
Best hyperparameters: {'max_depth': None, 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 200}
max_depth: None: the nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.
min_samples_leaf: 2: the minimum number of samples required to be at a leaf node. Setting it to 2 helps prevent overfitting by ensuring that leaf nodes have at least 2 samples.
min_samples_split: 2: the minimum number of samples required to split an internal node. Setting it to 2 means that a node must have at least 2 samples to be considered for splitting.
n_estimators: 200: the number of trees in the forest. Setting it to 200 means that the Random Forest model consists of 200 decision trees.

Confusion Matrix:
Class 0: 23 correct predictions, 0 misclassifications.
Class 1: 24 correct predictions, 0 misclassifications.
Class 2: 25 correct predictions, 1 misclassification (predicted as Class 4).
Class 3: 13 correct predictions, 0 misclassifications.
Class 4: 17 correct predictions, 0 misclassifications.
Class 5: 11 correct predictions, 0 misclassifications.
Class 6: 19 correct predictions, 1 misclassification (predicted as Class 4).
Class 7: 22 correct predictions, 0 misclassifications.
Class 8: 19 correct predictions, 0 misclassifications.
Class 9: 19 correct predictions, 0 misclassifications.

Hight precision, recall, F1 score indicate the model performs well in terms of correctly identifying positive cases and minimizing false positive. Most classes are predicted correctly with few misclassifications.



Numerical
I use image dataset called “boneage” (bone age) which contains 629 images, also a csv file with 3 columns: id, boneage, male(true or false).

Linear model:
Initially, I get the error message: valueError: With n_samples=0, test_size=0.2 and train_size=None, the resulting train set will be empty. Adjust any of the aforementioned parameters. Either X or y (or both) are empty at the time of the split.

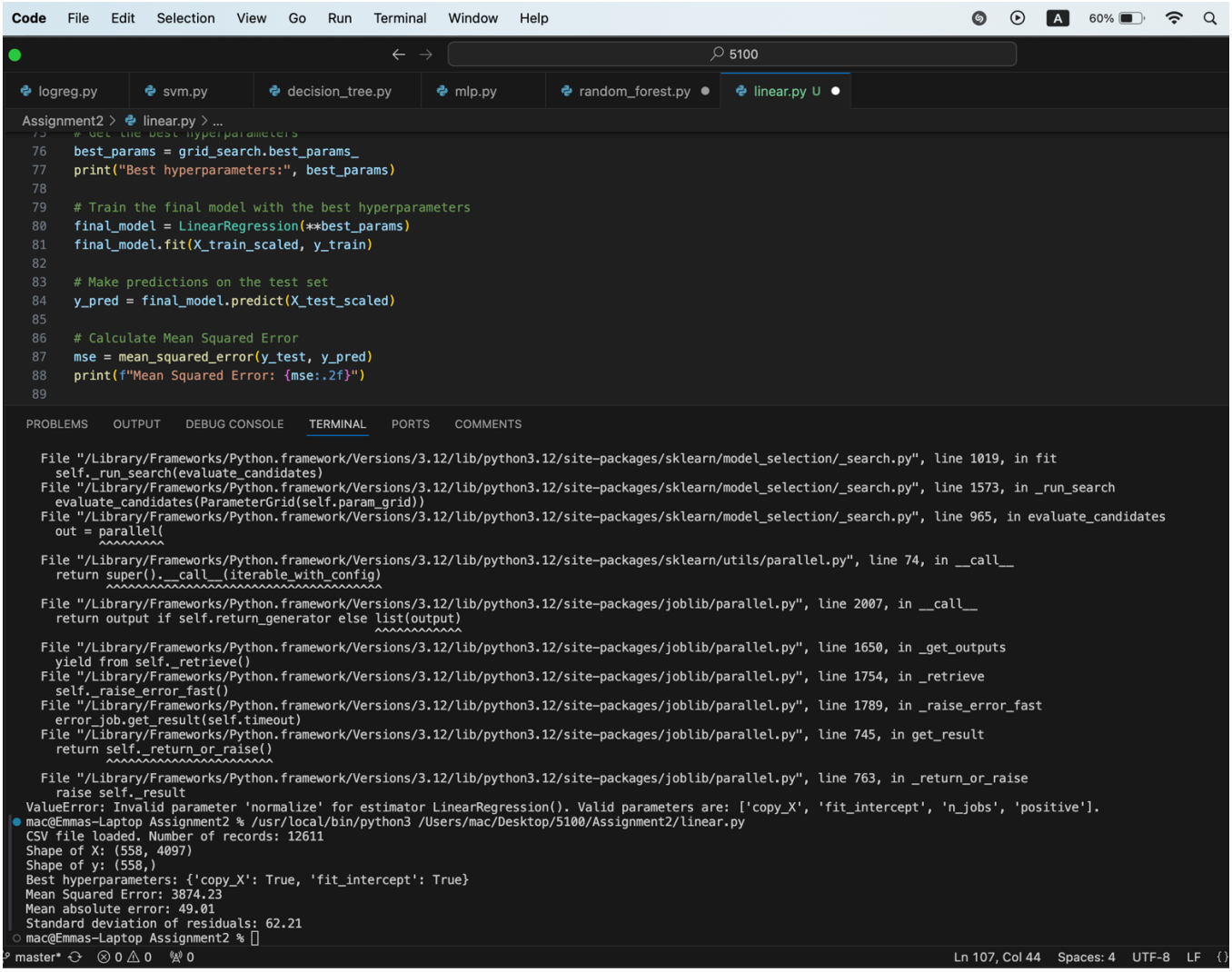
How to fix: handle the case where the image IDs range from 1377 to 2005 and ensure it works even if some IDs do not have corresponding images.

```
if 1377 <= image_id <= 2005:  # Ensure the image ID is within the specified range
    image_path = f'boneage/{image_id}.png'
    if os.path.exists(image_path):
        image_features.append(load_image(image_path))
        valid_indices.append(index)
```

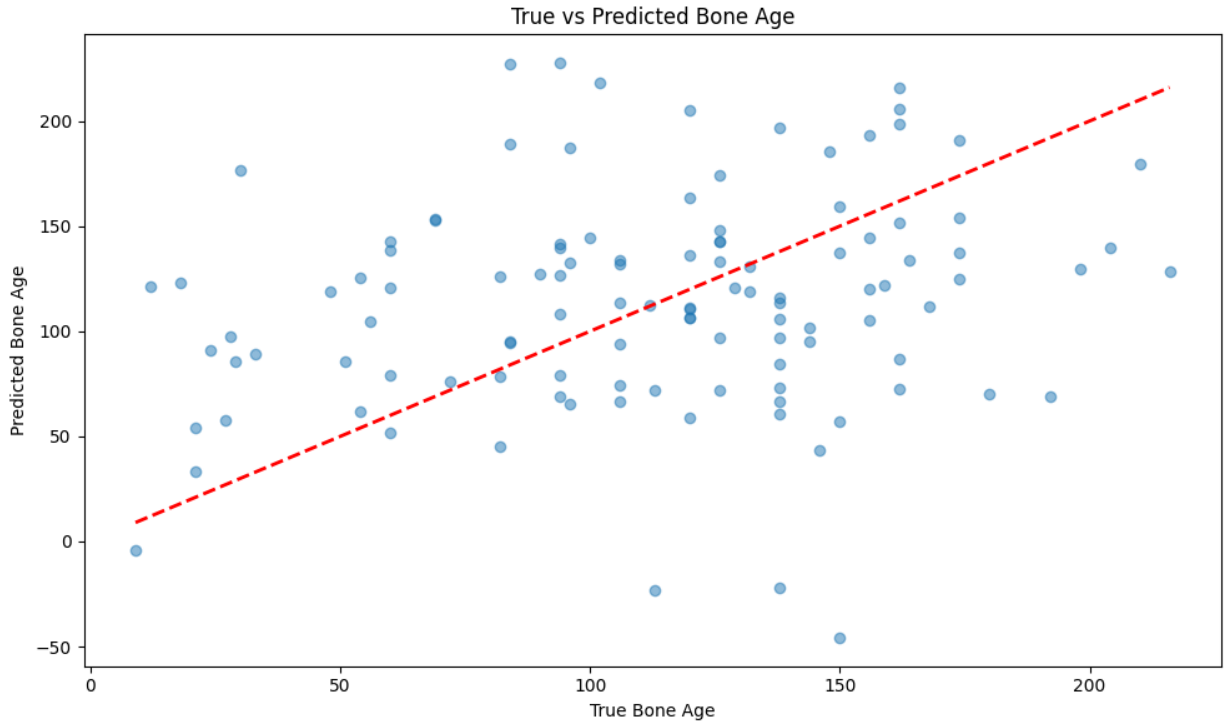
Outcome:
Shape of X: (558, 4097)
Shape of y: (558,)
Shape of X: The feature matrix X has 558 samples (rows) and 4097 features (columns). The features include the gender and the flattened image data.
Shape of y: The target vector y has 558 samples. This indicates that after filtering for valid images, 558 samples remain.

Best hyperparameters: {'copy_X': True, 'fit_intercept': True}
copy_X: True: This means that the input X will be copied during fitting. This is to avoid modifying the original data.
fit_intercept: True: This means that the model will calculate the intercept for the linear regression.

Mean Squared Error: 3874.23

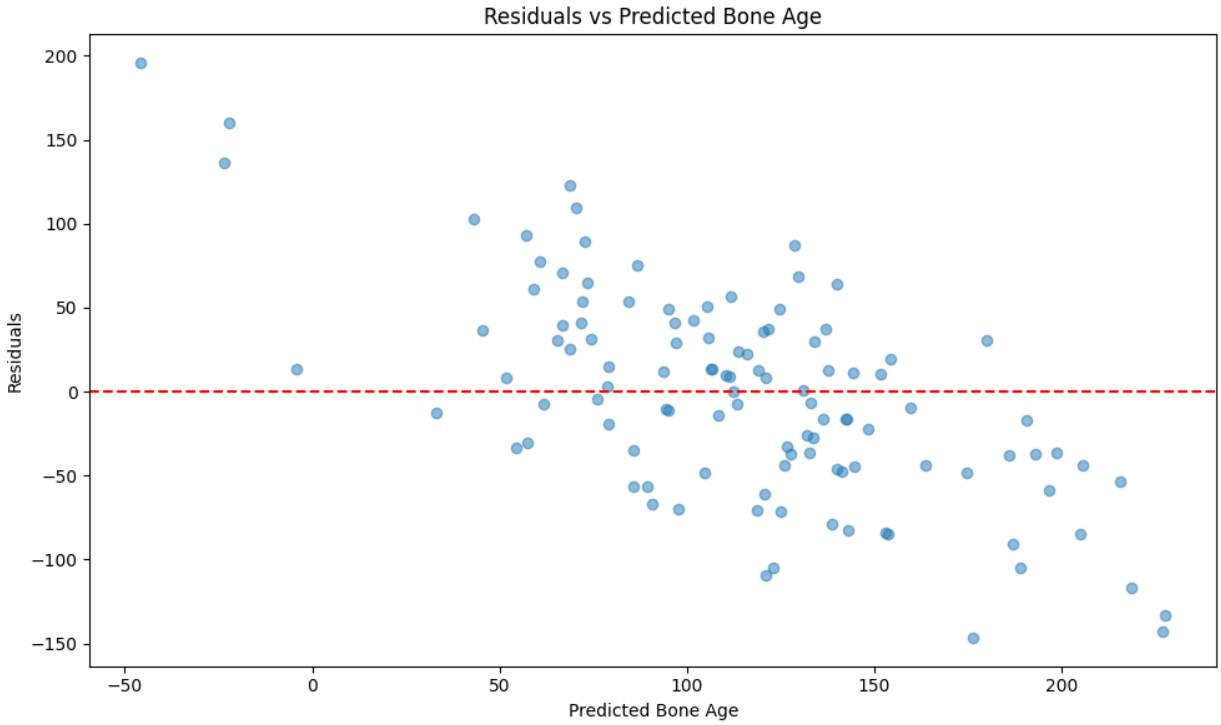

Code File Edit Selection View Go Run Terminal Window Help
60%
5100
logreg.py
svm.py
decision_tree.py
mlp.py
random_forest.py
linear.py U
Assignment2 > linear.py > ...
76 best_params = grid_search.best_params_
77 print("Best hyperparameters:", best_params)
79 # Train the final model with the best hyperparameters
80 final_model = LinearRegression(**best_params)
81 final_model.fit(X_train_scaled, y_train)
83 # Make predictions on the test set
84 y_pred = final_model.predict(X_test_scaled)
86 # Calculate Mean Squared Error
87 mse = mean_squared_error(y_test, y_pred)
88 print(f"Mean Squared Error: {mse:.2f}")
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
File "/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages/sklearn/model_selection/_search.py", line 1019, in fit
self._run_search(evaluate_candidates)
File "/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages/sklearn/model_selection/_search.py", line 1573, in _run_search
evaluate_candidates(ParameterGrid(self.param_grid))
File "/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages/sklearn/model_selection/_search.py", line 965, in evaluate_candidates
out = parallel(
File "/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages/sklearn/utils/parallel.py", line 74, in __call__
return super().__call__(iterable_with_config)
File "/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages/joblib/parallel.py", line 2007, in __call__
return output if self.return_generator else list(output)
File "/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages/joblib/parallel.py", line 1650, in _get_outputs
yield from self._retrieve()
File "/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages/joblib/parallel.py", line 1754, in _retrieve
self._raise_error_fast()
File "/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages/joblib/parallel.py", line 1789, in _raise_error_fast
error_job.get_result(self.timeout)
File "/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages/joblib/parallel.py", line 745, in get_result
return self._return_or_raise()
File "/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages/joblib/parallel.py", line 763, in _return_or_raise
raise self._result
ValueError: Invalid parameter 'normalize' for estimator LinearRegression(). Valid parameters are: ['copy_X', 'fit_intercept', 'n_jobs', 'positive'].
mac@Emmas-Laptop Assignment2 % /usr/local/bin/python3 /Users/mac/Desktop/5100/Assignment2/linear.py
CSV file loaded. Number of records: 12611
Shape of X: (558, 4097)
Shape of y: (558,)
Best hyperparameters: {'copy_X': True, 'fit_intercept': True}
Mean Squared Error: 3874.23
Mean absolute error: 49.01
Standard deviation of residuals: 62.21
mac@Emmas-Laptop Assignment2 %
master* 0 0 0
Ln 107, Col 44 Spaces: 4 UTF-8 LF

The relationship between true bone age and predicted bone age:



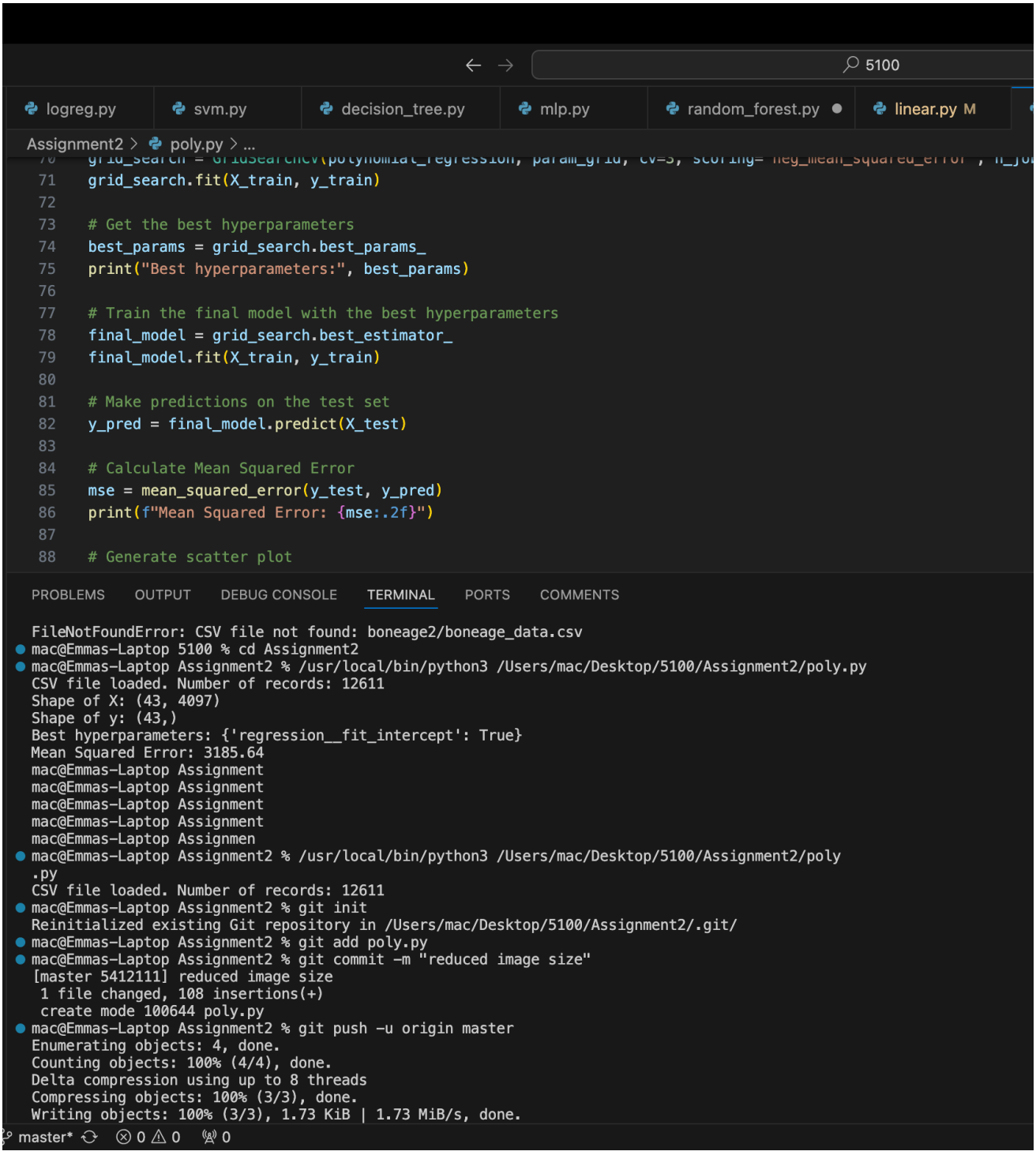
The model shows a positive correlation between true and predicted bone ages, but with significant variability in prediction accuracy.

The residuals plotted against the predicted bone age:

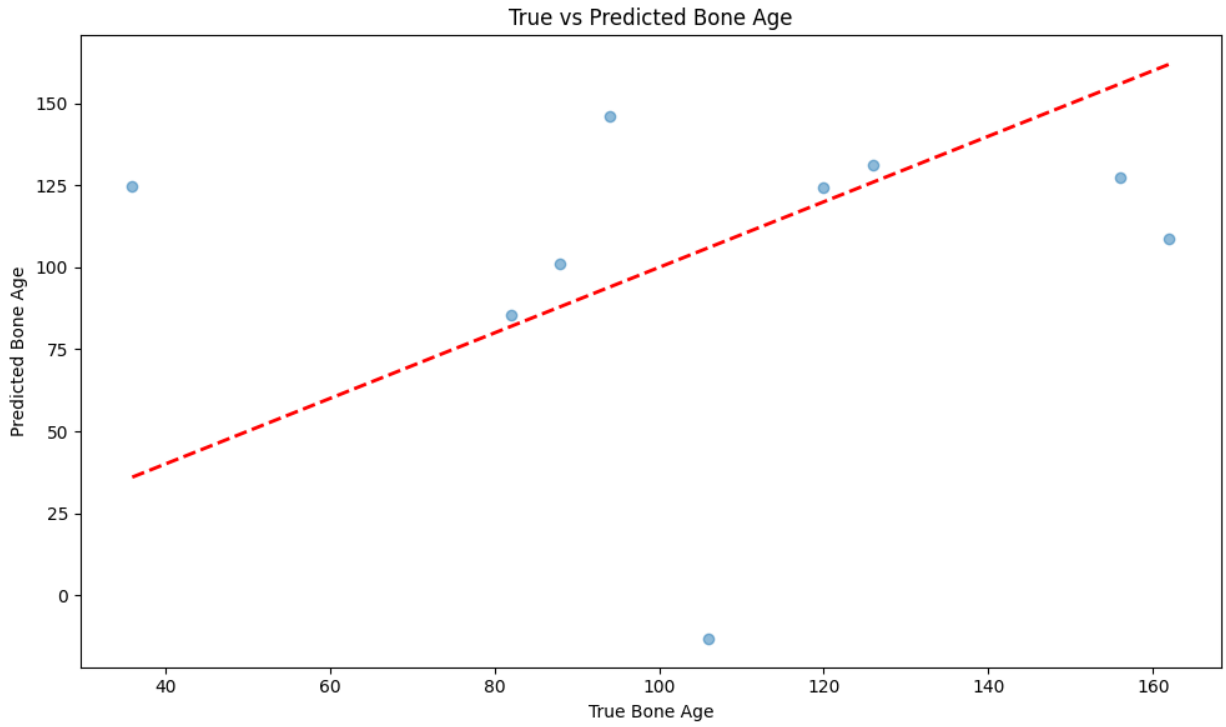


The model is unbiased but has significant prediction errors for some cases. There's no clear pattern or trend, which is generally good for a residual plot.

Polynomial:
Reduce the number of images to 50 to avoid excessive memory usage issue. Also set a fixed degree (=2) to reduce complexity, and reduce cv to 3.

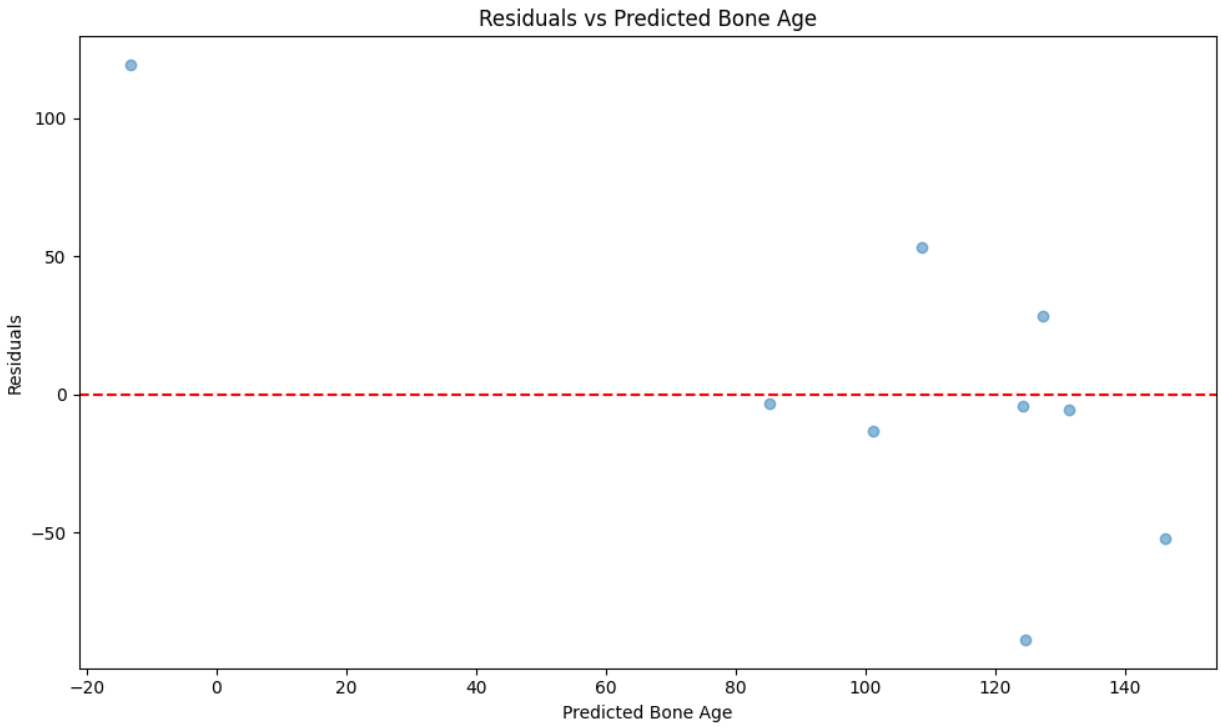


The relationship between true bone age and predicted bone age:



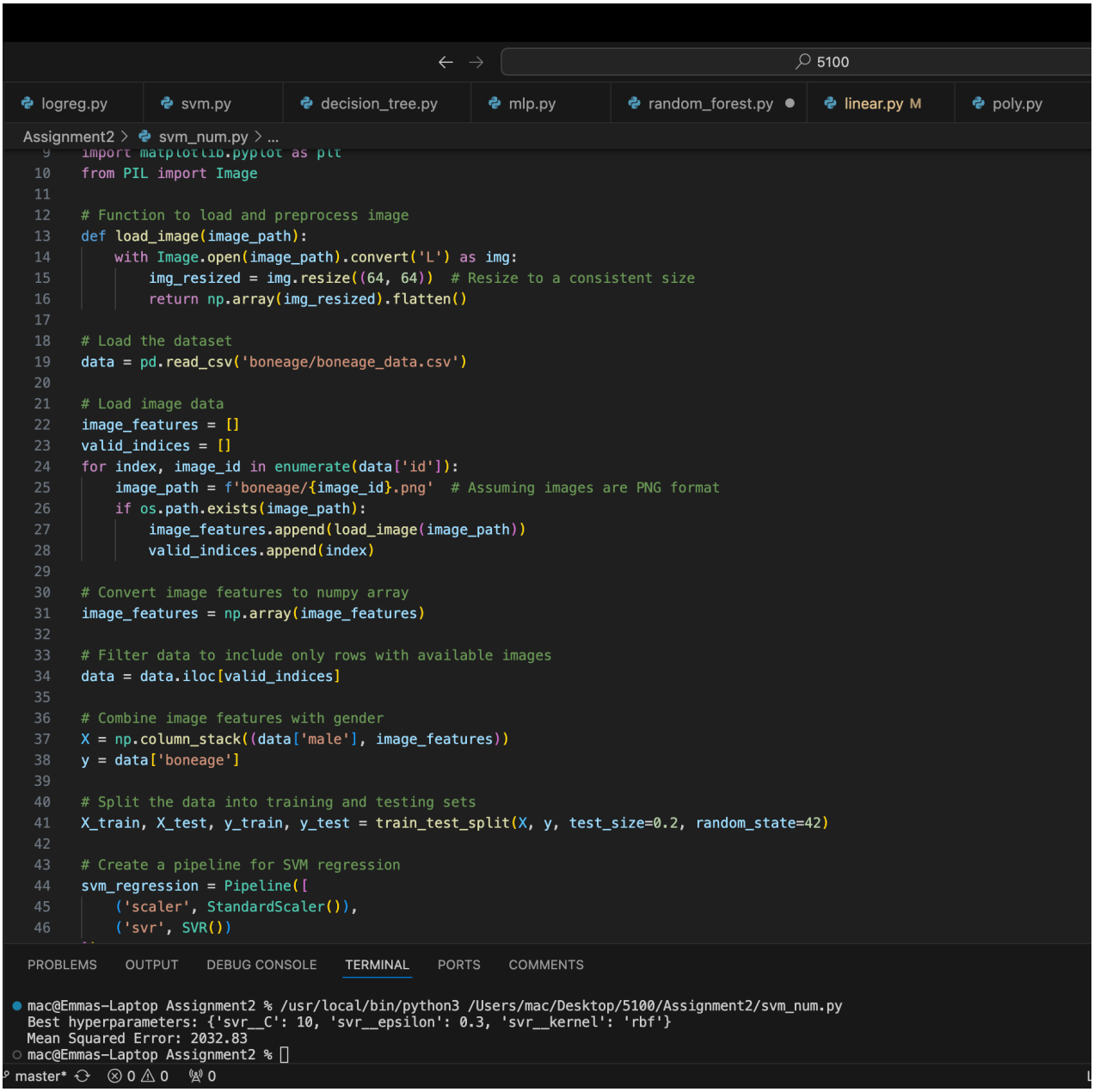
The bone age prediction model generally performs reasonably well, with predictions often close to true values. However, the presence of outliers and scatter indicates that the model can sometimes make significant errors. The small sample size limits how much it can generalize about the model's overall performance.

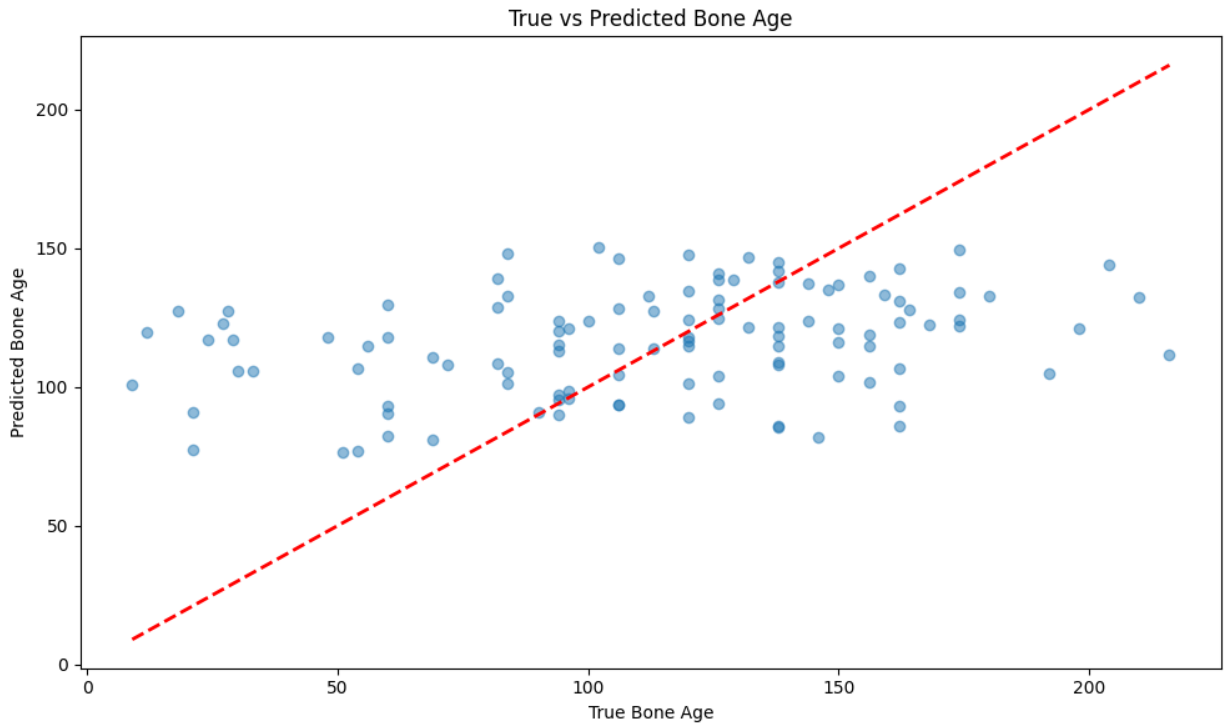
The residuals plotted against the predicted bone age:



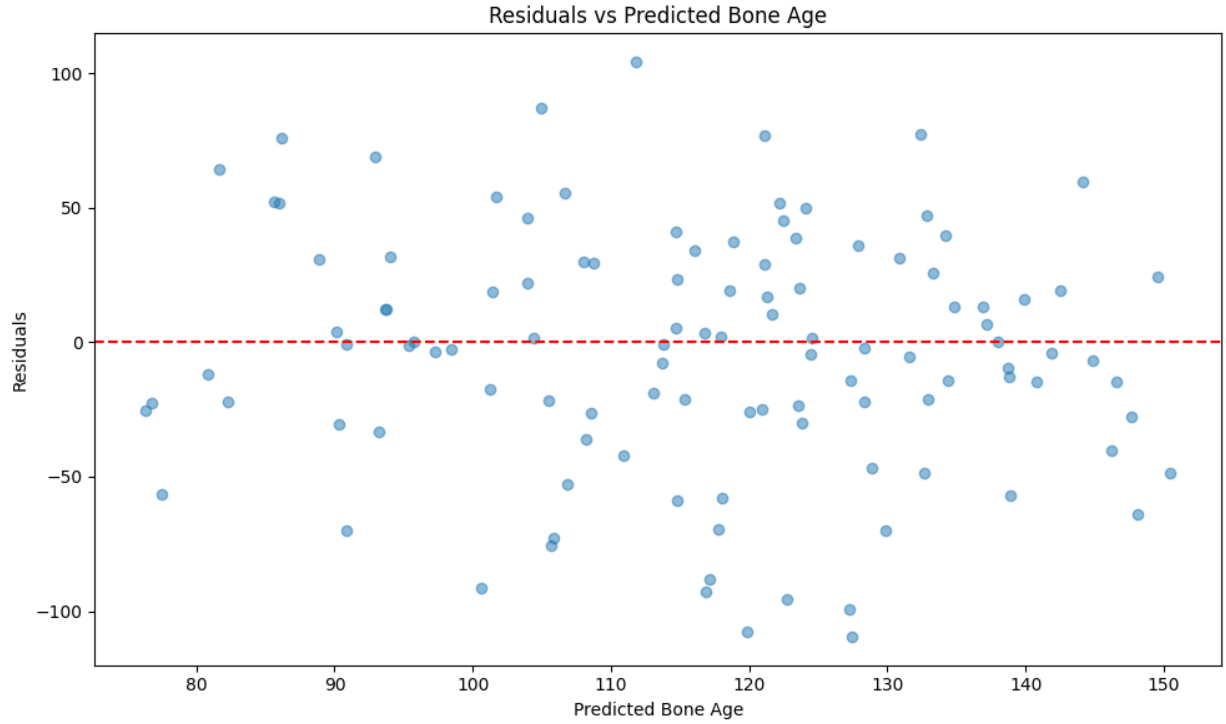
The model is unbiased but has significant prediction errors for some cases. The patten of residuals is not clear.

Support vector machine:





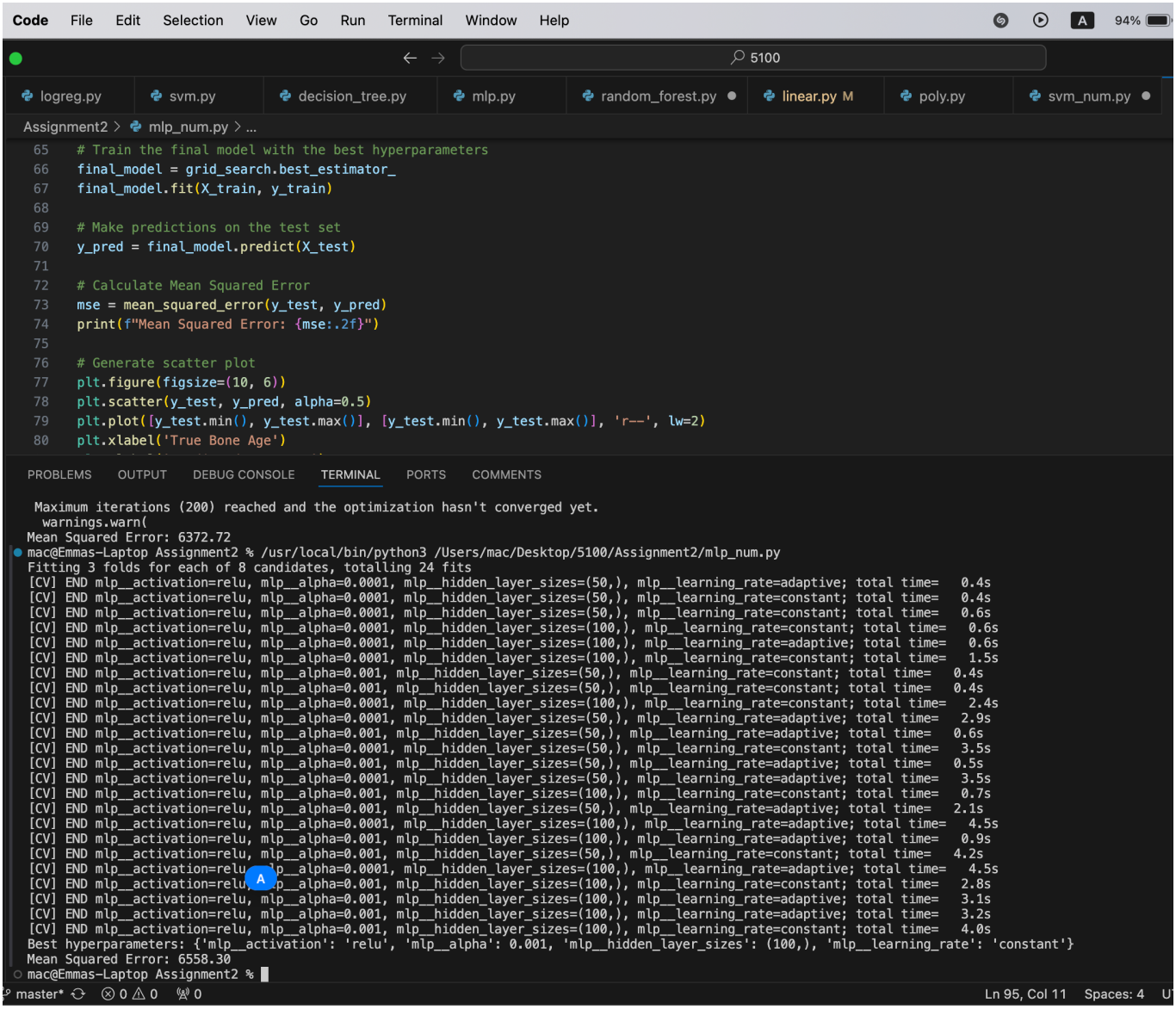
The bone age prediction model has a positive correlation with true ages but exhibits considerable variability in its predictions. The model's accuracy seems to decrease for higher bone ages.

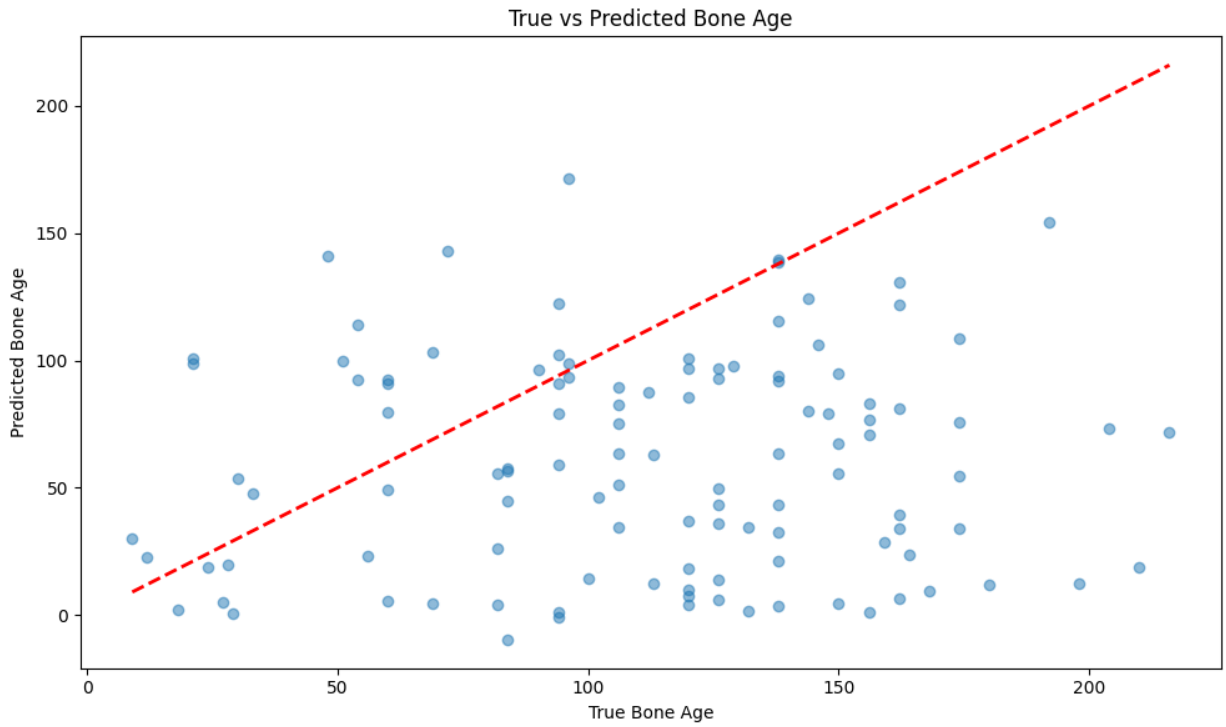


The points are scattered around the zero line. It indicates that the model is not systematically over or under-predicting across all bone ages.
There's no clear trend or curve in the residuals as the predicted bone age increases. The model is not systematically biased for different age ranges. The spread of residuals seems fairly consistent across the range of predicted ages.

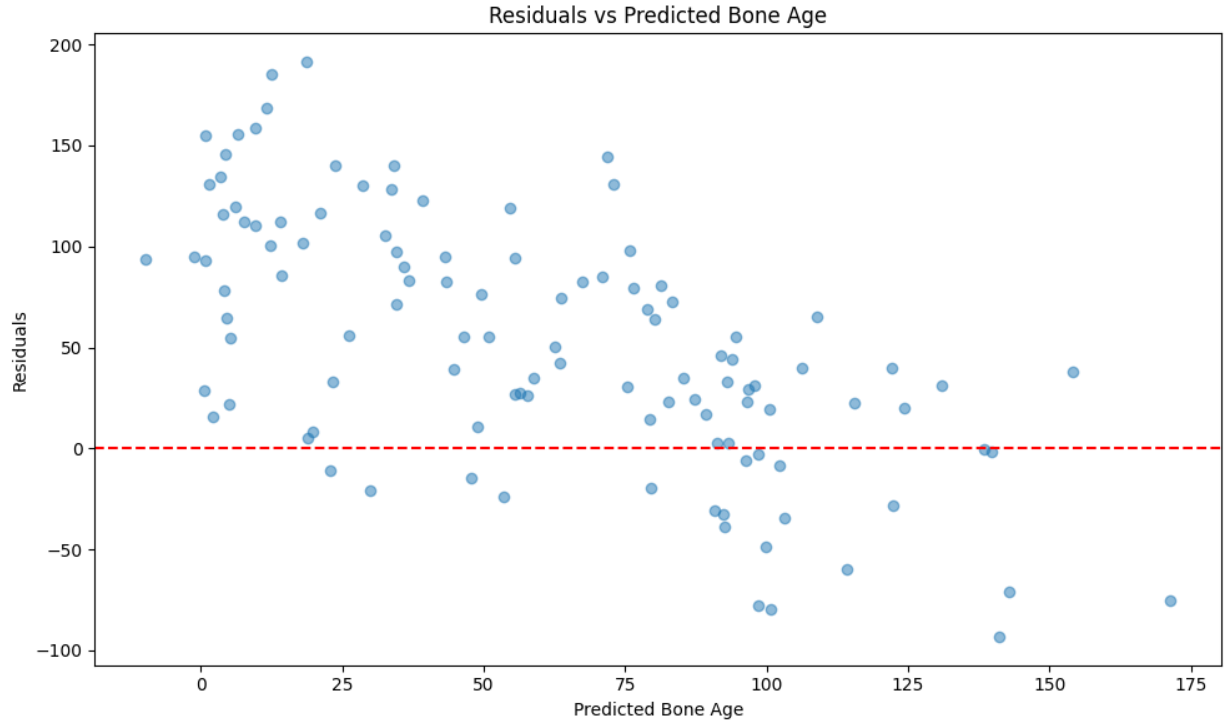
Multi-layer perceptron:
Initially, the model has error message: Stochastic Optimizer: Maximum iterations (1000) reached and the optimization hasn't converged yet.

How to fix: Simplify the parameter grid to reduce the number of combinations and speed up the training process.
Increase Verbosity: Add verbosity to the GridSearchCV to monitor the progress and identify if it's getting stuck.





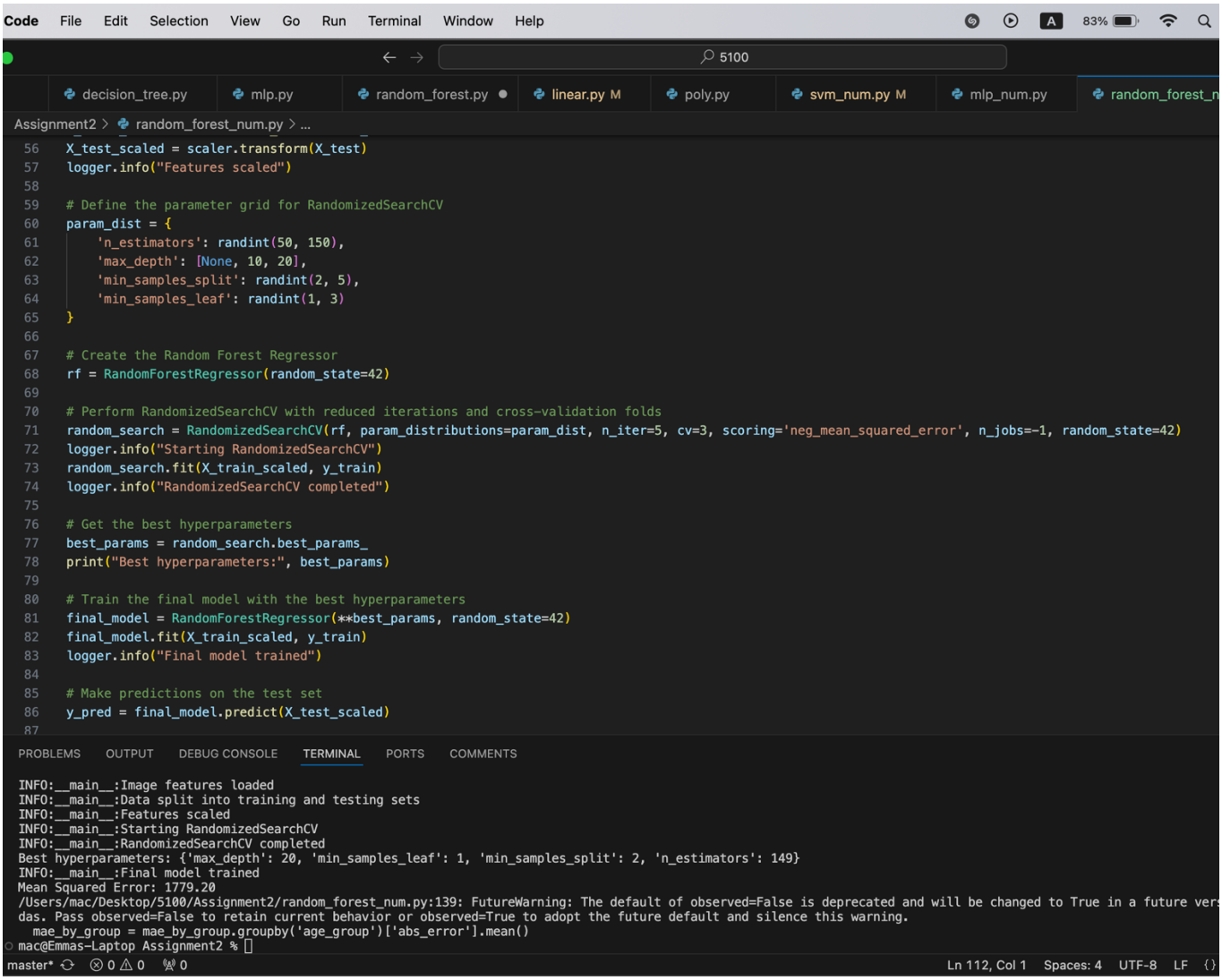
Many points cluster around red dashed line (perfect predictions), indicating that the model has learned some useful patterns. The spread seems to increase for higher bone ages, the model may be less accurate for older individuals.

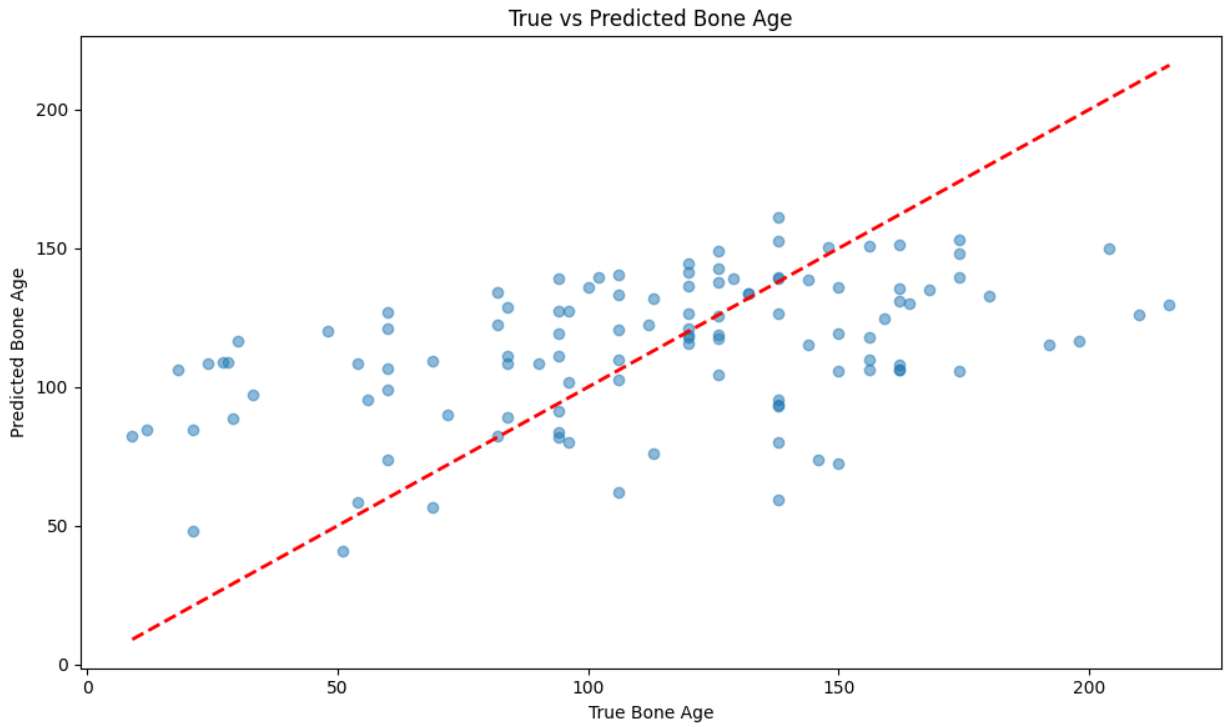


There's a noticeable pattern in the residuals. The spread seems to decrease as the predicted bone age increases. This suggests heteroscedasticity - the variance of residuals is not constant across all predicted values. For lower predicted ages (0-50 months), there's a tendency for positive residuals, the model often underestimates bone age for younger subjects.
For higher predicted ages (100+ months), there's a slight tendency for negative residuals, suggesting mild overestimation for older subjects.

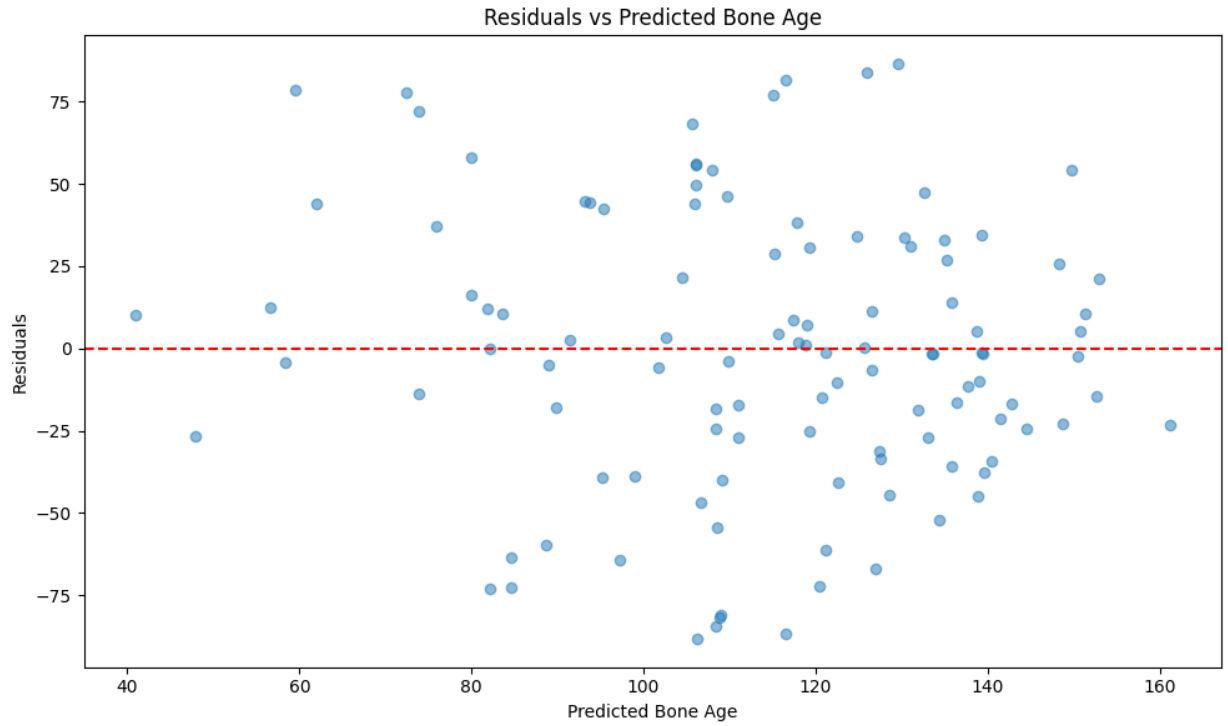
Random forest regressor:

This model has the lowest mean squared error.





The spread seems relatively consistent across different ages. There's no obvious bias towards over or under-prediction for specific age ranges.



There's no clear trend or curve in the residuals as the predicted bone age increases.
The spread of residuals seems fairly constant across the range of predicted ages.