CSCS 460 – Machine Learning

Assignment 3 (Task 3) **Deadline: Friday 5th May 2023 11:59 PM**

Submission Guidelines:

- Submit all three tasks together as a single zip file on Moodle. All three tasks should have separate folders. Submit actual code files and not the snapshots.
- Only one member of the group has to submit on Moodle.
- Mention Roll numbers in the zip file name e.g., 23123123_2351551.zip
- A viva will be conducted during office hours after the submission.

Task 1 Submission Checklist

- Code to find best hyperparameters for SGDRegressor using GridSearchCV.
- Code to train SGDRegressor on best hyperparameters on training split.
- Compute R^2 and MSE for predictions on test split using the trained SGDRegressor.
- Code to find best hyperparameters for RidgeCV, ElasticNetCV, and LassoCV.
- Code to training Ridge, ElasticNet, and Lasso regression models using hyperparameters found above on training split.
- Compute R^2 and MSE for prediction on the test split using the trained Ridge, ElasticNet, and Lasso regression models.
- Submit the code files for your webcam script that uses the best performing model for predicting the age.

Task 2 Submission Checklist

- Code to train logistic regression classifier with “me” and “not me” labels along with performance output on test split.
- Code to train multinomial “oVr” logistic regression classifier along with performance output on test split.
- Updated webcam script that shows age, me/not me label with probability, expression label with probability on webcam video stream in real-time. Use the models that give you best performance on test split.
- Code for Gradio GUI that takes image as input and displays the label me/not me along with probability.

Task 3: Using TensorFlow Keras Functional Model API

Note: Do not use Sequential model API. Use Functional model API instead.

1) Implement Linear Regression model for your “age” prediction dataset using TensorFlow and train the model on your training split. Plot the model graphically using TensorFlow “plot_model” method.

a) Test the model using test split with appropriate metrics.
b) Use appropriate loss function and activation function/s.
c) Train the model for 50 epochs.
d) Use checkpointing to save the model on the epoch where your model has minimum loss based on the test split (See TensorFlow documentation on model checkpoint callback)

2) Implement binary-class classification using logistic regression with sigmoid activation for "me" vs "not me" labels in your dataset using TensorFlow. Train on training split. Plot the model graphically using TensorFlow "plot_model" method.
   a) Test the model on testing split with appropriate metrics.
   b) Use appropriate loss function and activation function/s.
   c) Train the model for 50 epochs.
   d) Use checkpointing to save the model on the epoch where your model has minimum loss based on the test split
3) Implement Multi-class classification using logistic regression with softmax activation for emotions labels in your dataset using TensorFlow. Train on training split. Plot the model graphically using TensorFlow "plot_model" method.
   a) Test the model on testing split with appropriate metrics.
   b) Use appropriate loss function and activation function/s.
   c) Train the model for 50 epochs.
   d) Use checkpointing to save the model on the epoch where your model has minimum loss based on the test split
4) Now that you have all three models, for each of the labels in the dataset, use them in your webcam script. Show the predicted age, binary labels with probability, and multi-class labels with probability.

Hint: You just need to make a few adjustments to Linear Regression model for "age" to convert it to a logistic regression model (sigmoid) and logistic regression model with multi-class classification (softmax)

Task 3 Submission Checklist

- Submit the code file/s that perform the required functionalities for task 3.

Some Helpful Libraries and Modules

- **Sklearn:** `LogisticRegression`, `classification_report`, `r2_score`, `SGDRegressor`, `GridSearchCV`, `RidgeCV`, `ElasticNetCV`, `LassoCV`, `Ridge`, `ElasticNet`, `Lasso`
- **Numpy:** `flatten`, `reshape`
- **Cv2:** `resize`, `imread`, `imshow`, `VideoCapture`, `waitKey`, `destroyAllWindows`, `putText`
- **Gradio:** `Interface`

- **TensorFlow:** plot_model, layers.Dense, layers.Flatten, layers.Input, Model, compile, fit, predict