**Practical 5**

**K-Fold Cross Validation**

**5 a)**

**Aim:** Evaluating feed forward deep network for regression using KFold cross validation.

**Description:**

1. **Feedforward Deep Network**: A feedforward deep network, also known as a feedforward neural network, is a type of artificial neural network where connections between units do not form cycles. Information flows in one direction, from input to output layers, without any feedback loops. This architecture is commonly used for regression tasks, where the goal is to predict a continuous output value based on input features.
2. **Regression**: Regression is a type of supervised learning task where the goal is to predict a continuous output variable based on one or more input features. In the context of this code, the neural network model is trained to predict housing prices (a continuous variable) based on various features such as the number of rooms, crime rate, and accessibility to highways.
3. **KFold Cross-Validation**: KFold cross-validation is a technique used to assess the performance of a machine learning model. It involves splitting the dataset into K folds (or subsets) of approximately equal size. The model is trained K times, each time using K-1 folds for training and one fold for validation. This process allows for a more robust estimation of the model's performance compared to a single train-test split.
4. **Neural Network Architecture**: The neural network architecture used in this code consists of multiple layers of neurons (Dense layers) organized in a sequential manner. Each layer applies a transformation to the input data using activation functions like ReLU (Rectified Linear Unit). The final layer outputs a continuous value, making it suitable for regression tasks.
5. **Pipeline Construction**: A scikit-learn **Pipeline** is constructed to sequentially apply a list of transforms and a final estimator. In this case, the transforms include standardization (**StandardScaler**) to scale the input features and the final estimator is the Keras neural network model (**KerasRegressor**). This pipeline ensures consistent preprocessing of data during cross-validation and simplifies the evaluation process.

**Code:**

# !pip install keras (2.15.0)

# !pip install scikit\_learn

# !pip install scikeras

import pandas as pd

from keras.models import Sequential

from keras.layers import Dense

# from keras.wrappers.scikit\_learn import KerasRegressor

from scikeras.wrappers import KerasRegressor

from sklearn.model\_selection import cross\_val\_score, KFold

from sklearn.preprocessing import StandardScaler

from sklearn.pipeline import Pipeline

from sklearn.neural\_network import MLPRegressor

dataframe = pd.read\_csv("MscIT\Semester 4\Deep\_Learning\Practical05\housing.csv")

# dataframe = pd.read\_csv("/content/housing.csv")

dataset = dataframe.values

# Print the shape of dataset to verify the number of features and samples

print("Shape of dataset:", dataset.shape)

# Ensure correct slicing for features and target variable

X = dataset[:, :-1] # Select all columns except the last one as features

Y = dataset[:, -1] # Select the last column as target variable

def wider\_model():

model = Sequential()

model.add(Dense(15, input\_dim=13, kernel\_initializer='normal', activation='relu'))

# model.add(Dense(20, input\_dim=13, kernel\_initializer='normal', activation='relu'))

model.add(Dense(13, kernel\_initializer='normal', activation='relu'))

model.add(Dense(1, kernel\_initializer='normal'))

model.compile(loss='mean\_squared\_error', optimizer='adam')

return model

estimators = []

estimators.append(('standardize', StandardScaler()))

estimators.append(('mlp', KerasRegressor(build\_fn=wider\_model, epochs=10, batch\_size=5)))

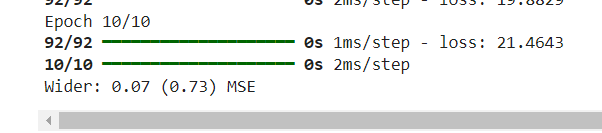
pipeline = Pipeline(estimators)

kfold = KFold(n\_splits=10)

results = cross\_val\_score(pipeline, X, Y, cv=kfold)

print("Wider: %.2f (%.2f) MSE" % (results.mean(), results.std()))

**Output:**



**Learning:**

1. **Importing and Installing Libraries**: The code starts by installing necessary libraries like Keras, scikit-learn, and scikeras using pip. These libraries are essential for building and evaluating the neural network model.
2. **Data Loading and Preprocessing**: The dataset is loaded using pandas from a CSV file named "housing.csv". The dataset is then split into input features (X) and target variable (Y). Standardization is performed on the input features using **StandardScaler** from scikit-learn to ensure all features have a mean of 0 and a standard deviation of 1.
3. **Neural Network Architecture**: A wider feedforward neural network model is defined using Keras. It consists of an input layer with 13 neurons (matching the number of features), a hidden layer with 15 neurons and ReLU activation function, another hidden layer with 13 neurons and ReLU activation function, and an output layer with 1 neuron for regression. The model is compiled with mean squared error loss function and Adam optimizer.
4. **Pipeline Construction**: A scikit-learn **Pipeline** is constructed to sequentially apply a list of transforms and a final estimator. In this case, the transforms include standardization (**StandardScaler**), followed by the Keras neural network model (**KerasRegressor**). This pipeline ensures consistent preprocessing of data during cross-validation.
5. **Cross-Validation and Model Evaluation**: Cross-validation is performed using scikit-learn's **cross\_val\_score** with 10 folds. During each fold, the pipeline is trained and evaluated on different subsets of the dataset. The mean squared error (MSE) is computed for each fold, and the average MSE across all folds is calculated and printed as the evaluation metric for the wider neural network model.
6. **Comparison with Other Models**: The code provides a benchmark for evaluating the performance of the wider neural network model. Other models or configurations could be tested similarly by defining different neural network architectures or adjusting hyperparameters, and their performance can be compared to determine the most effective model for the given regression task.

**5 b)**

**Aim:** Evaluating feed forward deep network for multiclass Classification using KFold crossvalidation.

**Description:**

1. Feedforward Deep Network: The term "feedforward deep network" refers to a type of artificial neural network where connections between units do not form cycles. In other words, information flows in one direction, from input to output layers. This architecture is commonly used for tasks such as classification, regression, and function approximation.
2. Multiclass Classification: Multiclass classification is a supervised learning task where the goal is to assign one of multiple class labels to each input sample. In this context, each sample can belong to only one class, and the goal is to predict the correct class label for new, unseen data points.
3. KFold Cross-Validation: KFold cross-validation is a technique used to assess the performance of a machine learning model. It involves splitting the dataset into K folds (or subsets) of approximately equal size. The model is trained K times, each time using K-1 folds for training and one fold for validation. This process allows for a more robust estimation of the model's performance compared to a single train-test split.
4. Label Encoding: Label encoding is a preprocessing step often used in machine learning for converting categorical labels into numeric format. In the context of multiclass classification, it involves mapping each class label to a unique integer identifier. This allows the model to understand and process the class labels during training.
5. One-Hot Encoding: One-hot encoding is another preprocessing step used in machine learning, particularly for multiclass classification tasks. It involves converting categorical variables into binary vectors, where each vector represents a unique class label. In this encoding scheme, a binary value of 1 is assigned to the position corresponding to the class label, while all other positions are set to 0. This representation is useful for training neural networks, as it ensures that class labels are represented in a meaningful and consistent way.
6. Model Evaluation Metrics: In the context of evaluating a deep neural network for multiclass classification, various evaluation metrics can be used to assess the model's performance. Common metrics include accuracy, precision, recall, F1 score, and confusion matrix. These metrics provide insights into different aspects of the model's performance, such as its ability to correctly classify instances of each class and its overall predictive accuracy.

**Code:**

# 5B. Evaluating feed forward deep network for multiclass Classification using KFold cross-validation.

# !pip install scikeras

# !pip install np\_utils

# loading libraries

import pandas

from keras.models import Sequential

from keras.layers import Dense

from scikeras.wrappers import KerasClassifier

from tensorflow.keras.utils import to\_categorical

from sklearn.model\_selection import cross\_val\_score

from sklearn.model\_selection import KFold

from sklearn.preprocessing import LabelEncoder

# loading dataset

df = pandas.read\_csv('/content/flowers.csv', header=None) #remove , header=None if dataset contains column name

print(df)

# splitting dataset into input and output variables

X = df.iloc[:, 0:4].astype(float)

y = df.iloc[:, 4]

# print(X)

# print(y)

# encoding string output into numeric output

encoder = LabelEncoder()

encoder.fit(y)

encoded\_y = encoder.transform(y)

print(encoded\_y)

dummy\_Y = to\_categorical(encoded\_y)

print(dummy\_Y)

def baseline\_model():

# create model

model = Sequential()

model.add(Dense(8, input\_dim=4, activation='relu'))

model.add(Dense(3, activation='softmax'))

# Compile model

model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

return model

estimator = baseline\_model()

estimator.fit(X, dummy\_Y, epochs=100, shuffle=True)

action = estimator.predict(X)

for i in range(25):

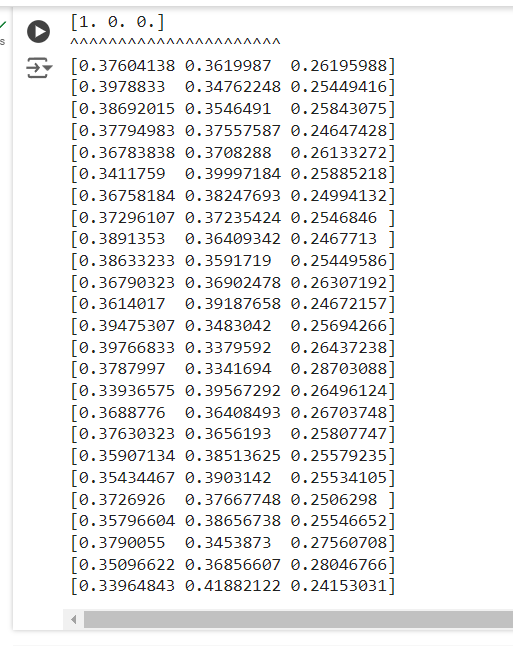
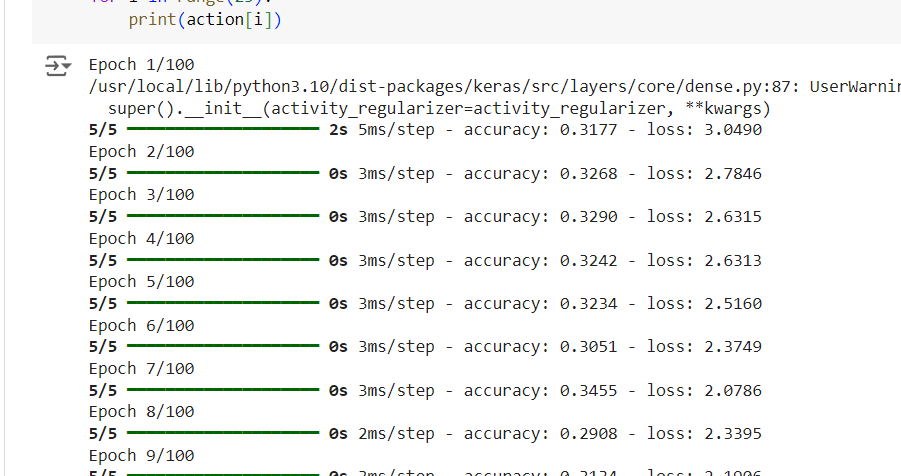
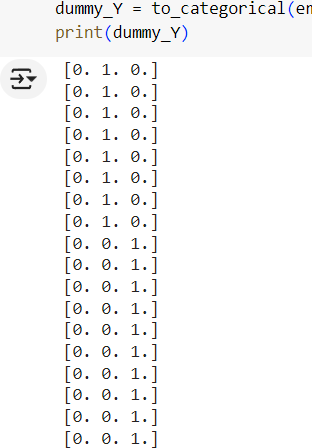
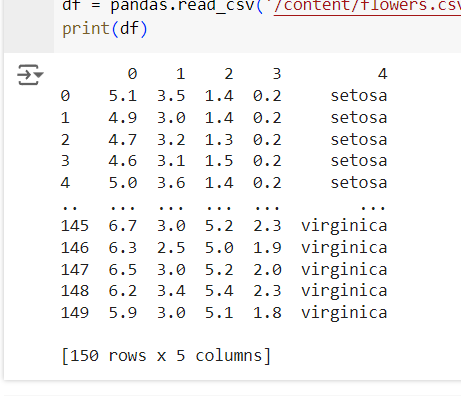
print(dummy\_Y[i])

print('^^^^^^^^^^^^^^^^^^^^^^')

for i in range(25):

print(action[i])

**Output:**

****

**Learning:**

1. **Importing and Installing Libraries**: The code begins with the installation of required libraries using pip, such as **scikeras** and **np\_utils**, which are necessary for building and evaluating the neural network model.
2. **Data Loading and Preprocessing**: The dataset, presumably containing information about flowers, is loaded into a pandas DataFrame. The dataset is then split into input features (X) and target variable (y). Additionally, the target variable is encoded from string labels into numerical format using **LabelEncoder**, and one-hot encoding is applied using **to\_categorical** to prepare it for multiclass classification.
3. **Neural Network Architecture**: A baseline feedforward neural network model is defined using Keras. It consists of an input layer, a hidden layer with 8 neurons and ReLU activation function, and an output layer with 3 neurons and softmax activation function for multiclass classification. The model is compiled with categorical cross-entropy loss function and Adam optimizer.
4. **Model Training**: The defined neural network model is trained on the input features (X) and the one-hot encoded target variable (dummy\_Y) for 100 epochs. The **fit** function is used to train the model, with shuffling enabled to ensure randomness in the training data.
5. **Model Evaluation**: After training, the model's predictions are generated using the **predict** function on the input data (X). The predictions are then printed alongside the corresponding actual target values for comparison. This step allows for a qualitative assessment of the model's performance on the training data.
6. **Further Analysis**: The code also prints the actual target values and predicted values for the first 25 instances in the dataset, allowing for visual inspection of the model's behavior. This analysis can provide insights into the model's ability to correctly classify instances and identify any potential issues or patterns in its predictions.