p229278

April 26, 2024

0.1 Roll No : P22-9278

0.2 Name : Muhammad Shafeen

0.3 Lab Task : 11

0.3.1 Importing Librarires

[ ]:
```
import numpy as np
import pandas as pd # for dealing with dataframes
from tensorflow.keras import Sequential # for creating a sequential model
from tensorflow.keras.layers import Dense # for creating layers in the model
import matplotlib.pyplot as plt # for plotting
import seaborn as sns # for plotting
from sklearn.preprocessing import StandardScaler, OneHotEncoder # for␣
 ↪preprocessing
from sklearn.impute import SimpleImputer # for preprocessing
from sklearn.compose import ColumnTransformer # for preprocessing
from sklearn.pipeline import Pipeline # for preprocessing
from sklearn.model_selection import train_test_split # for splitting the dataset
from sklearn.neural_network import MLPClassifier
```

0.3.2 Reading Dataset

[ ]:
```
path="/home/shafeenkhan/Documents/My-all-programs--/Semester-4/Aritificial␣
 ↪Intelligence/LAB-11/titanic.csv"
df=pd.read_csv(path)
df
```

[ ]:
```
     PassengerId  Survived  Pclass  \
0              1         0       3
1              2         1       1
2              3         1       3
3              4         1       1
4              5         0       3
..           …         …     …
886          887         0       2
887          888         1       1
888          889         0       3
```

```
889          890         1       1
890          891         0       3

                                                  Name     Sex   Age  SibSp  \
0                              Braund, Mr. Owen Harris    male  22.0      1
1    Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                               Heikkinen, Miss. Laina  female  26.0      0
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                             Allen, Mr. William Henry    male  35.0      0
..                                                 …     …   …    …
886                              Montvila, Rev. Juozas    male  27.0      0
887                       Graham, Miss. Margaret Edith  female  19.0      0
888           Johnston, Miss. Catherine Helen "Carrie"  female   NaN      1
889                              Behr, Mr. Karl Howell    male  26.0      0
890                                Dooley, Mr. Patrick    male  32.0      0

     Parch            Ticket     Fare Cabin Embarked
0        0         A/5 21171   7.2500   NaN        S
1        0          PC 17599  71.2833   C85        C
2        0  STON/O2. 3101282   7.9250   NaN        S
3        0            113803  53.1000  C123        S
4        0            373450   8.0500   NaN        S
..     …               …      …   …      …
886      0            211536  13.0000   NaN        S
887      0            112053  30.0000   B42        S
888      2        W./C. 6607  23.4500   NaN        S
889      0            111369  30.0000  C148        C
890      0            370376   7.7500   NaN        Q

[891 rows x 12 columns]
```

0.3.3 Checking Null values

[ ]:
```
df.isnull().sum()
# df=df.ffill()
df.isnull().sum()
# df=df.dropna()
df.isnull().sum()
```

[ ]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
```

```
Ticket            0
Fare              0
Cabin           687
Embarked          2
dtype: int64
```

[ ]:
```
# sex_data = df["Sex"].values.reshape(-1, 1)

# encoder = OneHotEncoder()

# one_hot_encoded = encoder.fit_transform(sex_data).toarray()

# print(one_hot_encoded)
```

0.3.4 Dropping columns that we donot need

[ ]:
```
# one_hot_encoded_df = pd.DataFrame(one_hot_encoded)
# df["Sex"]=one_hot_encoded_df[0]
df=df.drop(columns=["Name","Ticket","Cabin","Embarked","SibSp","Parch"]).copy()
df
```

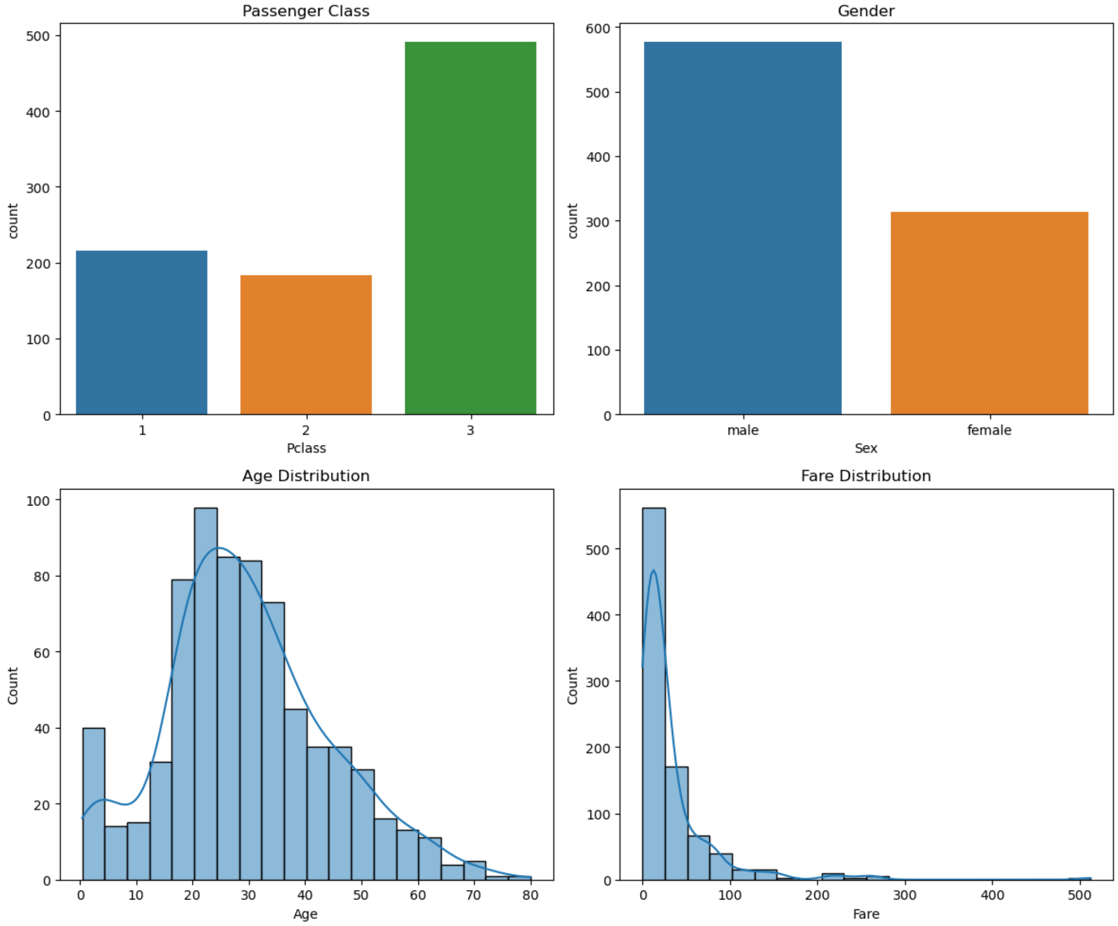
[ ]:
```
     PassengerId  Survived  Pclass     Sex   Age     Fare
0              1         0       3    male  22.0   7.2500
1              2         1       1  female  38.0  71.2833
2              3         1       3  female  26.0   7.9250
3              4         1       1  female  35.0  53.1000
4              5         0       3    male  35.0   8.0500
..           …         …       …       …     …      …
886          887         0       2    male  27.0  13.0000
887          888         1       1  female  19.0  30.0000
888          889         0       3  female   NaN  23.4500
889          890         1       1    male  26.0  30.0000
890          891         0       3    male  32.0   7.7500

[891 rows x 6 columns]
```

0.3.5 Plotting before normalization

[ ]:
```
# Set up the figure grid
fig, axes = plt.subplots(2, 2, figsize=(12, 10))

# Plot Pclass
sns.countplot(x='Pclass', data=df, ax=axes[0, 0])
axes[0, 0].set_title('Passenger Class')

# Plot Sex
sns.countplot(x='Sex', data=df, ax=axes[0, 1])
```

```python
axes[0, 1].set_title('Gender')

# Plot Age distribution
sns.histplot(data=df, x='Age', bins=20, kde=True, ax=axes[1, 0])
axes[1, 0].set_title('Age Distribution')

# Plot Fare distribution
sns.histplot(data=df, x='Fare', bins=20, kde=True, ax=axes[1, 1])
axes[1, 1].set_title('Fare Distribution')

# Adjust layout
plt.tight_layout()

# Show plots
plt.show()
```

```
/home/shafeenkhan/miniconda3/lib/python3.9/site-
packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na option is
deprecated and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/home/shafeenkhan/miniconda3/lib/python3.9/site-
packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na option is
deprecated and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



[ ]:
```
df
```

[ ]:
```
     PassengerId  Survived  Pclass     Sex   Age     Fare
0              1         0       3    male  22.0   7.2500
1              2         1       1  female  38.0  71.2833
2              3         1       3  female  26.0   7.9250
3              4         1       1  female  35.0  53.1000
4              5         0       3    male  35.0   8.0500
..           …         …       …     …     …
886          887         0       2    male  27.0  13.0000
887          888         1       1  female  19.0  30.0000
888          889         0       3  female   NaN  23.4500
889          890         1       1    male  26.0  30.0000
890          891         0       3    male  32.0   7.7500

[891 rows x 6 columns]
```

0.3.6 Normalization of the data

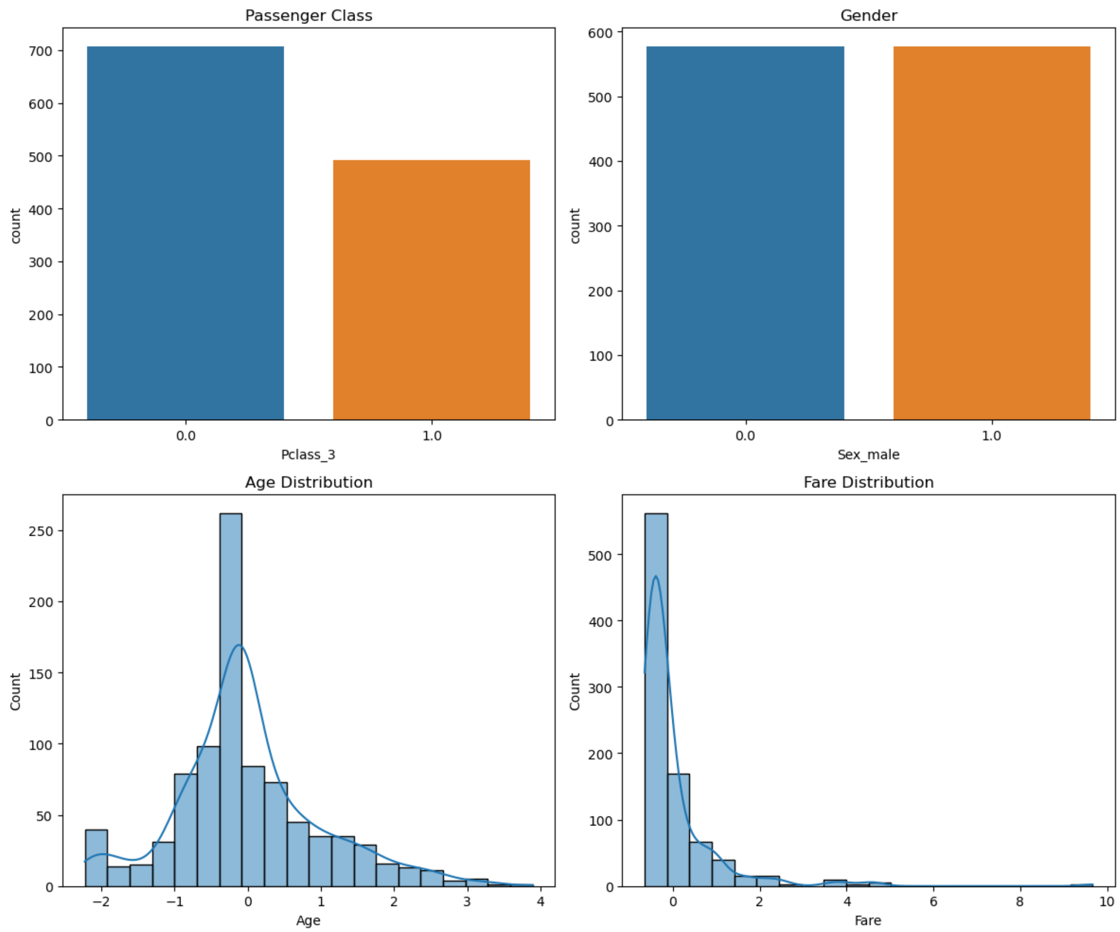
[ ]:
```
data = df[['Age', 'Fare', 'Pclass', 'Sex', 'Survived']].copy()

# Define preprocessing steps
numeric_features = [0, 1]  # Indices of numerical columns in data
numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),  # Fill missing values with␣
 ↪median
    ('scaler', StandardScaler())  # Scale data
])

categorical_features = [2, 3]  # Indices of categorical columns in data
categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='constant', fill_value='missing')),  #␣
 ↪Fill missing values with 'missing'
    ('onehot', OneHotEncoder())  # One-hot encode categorical variables
])

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)
    ])

# Apply preprocessing
X = preprocessor.fit_transform(data.drop(columns=['Survived']))
y = data['Survived']
```

0.3.7 Plotting After Normalization

[ ]:
```
X_df = pd.DataFrame(X, columns=['Age', 'Fare', 'Pclass_1', 'Pclass_2',␣
 ↪'Pclass_3', 'Sex_female', 'Sex_male'])

# Set up the figure grid
fig, axes = plt.subplots(2, 2, figsize=(12, 10))

# Plot Pclass
sns.countplot(x='Pclass_1', data=X_df, ax=axes[0, 0])
sns.countplot(x='Pclass_2', data=X_df, ax=axes[0, 0])
sns.countplot(x='Pclass_3', data=X_df, ax=axes[0, 0])
axes[0, 0].set_title('Passenger Class')

# Plot Sex
sns.countplot(x='Sex_female', data=X_df, ax=axes[0, 1])
sns.countplot(x='Sex_male', data=X_df, ax=axes[0, 1])
axes[0, 1].set_title('Gender')
```

```python
# Plot Age distribution
sns.histplot(data=X_df, x='Age', bins=20, kde=True, ax=axes[1, 0])
axes[1, 0].set_title('Age Distribution')

# Plot Fare distribution
sns.histplot(data=X_df, x='Fare', bins=20, kde=True, ax=axes[1, 1])
axes[1, 1].set_title('Fare Distribution')

# Adjust layout
plt.tight_layout()

# Show plots
plt.show()
```

```
/home/shafeenkhan/miniconda3/lib/python3.9/site-
packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na option is
deprecated and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/home/shafeenkhan/miniconda3/lib/python3.9/site-
packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na option is
deprecated and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



0.3.8 Splitting data testing and training

[ ]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
↪2,random_state=0)
```

0.3.9 Training Models on different architecture

0.3.10 WITH KERAS

[ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
models = [
    Sequential([ # Model 1 with 1 hidden layers
        Dense(32, activation='relu', input_shape=(7,)),
        Dense(32, activation='relu'),
        Dense(1, activation='sigmoid')
    ],name="Sequential_Titanic_model_1"),
```

```
    Sequential([ # Model 2 with 2 hidden layers
        Dense(64, activation='relu', input_shape=(7,)),
        Dense(100, activation='relu'),
        Dense(100, activation='relu'),
        Dense(1, activation='sigmoid')
    ],name="Sequential_Titanic_model_2"),
    Sequential([ # Model 3 with many hidden layers and more neurons
        Dense(10, activation='relu', input_shape=(7,)),
        Dense(20, activation='relu'),
        Dense(30, activation='relu'),
        Dense(40, activation='relu'),
        Dense(50, activation='relu'),
        Dense(60, activation='relu'),
        Dense(70, activation='relu'),
        Dense(80, activation='relu'),
        Dense(90, activation='relu'),
        Dense(1, activation='sigmoid')
    ],name="Sequential_Titanic_model_3"),
    Sequential([ # Model 4 with one hidden layer and many units
        Dense(100, activation='relu', input_shape=(7,)),
        Dense(1, activation='sigmoid')
    ],name="Sequential_Titanic_model_4"),
    # Sequential([
    #     Dense(3, activation='relu', input_shape=(7,)),
    #     Dense(4, activation='relu'),
    #     Dense(1, activation='sigmoid')
    # ],name="Sequential_Titanic_model_5"),
    # Sequential([
    #     Dense(9, activation='relu', input_shape=(7,)),
    #     Dense(8, activation='relu'),
    #     Dense(5, activation='relu'),
    #     Dense(1, activation='sigmoid')
    # ],name="Sequential_Titanic_model_6"),
    # Sequential([
    #     Dense(0.2,activation='relu',input_shape=(7,)),
    #     Dense(0.5,activation='relu'),
    #     Dense(0.4,activation='relu'),
    #     Dense(0.5,activation='relu'),
    #     Dense(0.01,activation='relu'),
    #     Dense(0.05,activation='relu'),
    #     Dense(1,activation='sigmoid'),],name="Sequential_Titanic_model_7"),
]

# Compiling and  training models
histories = []
for model in models:
```

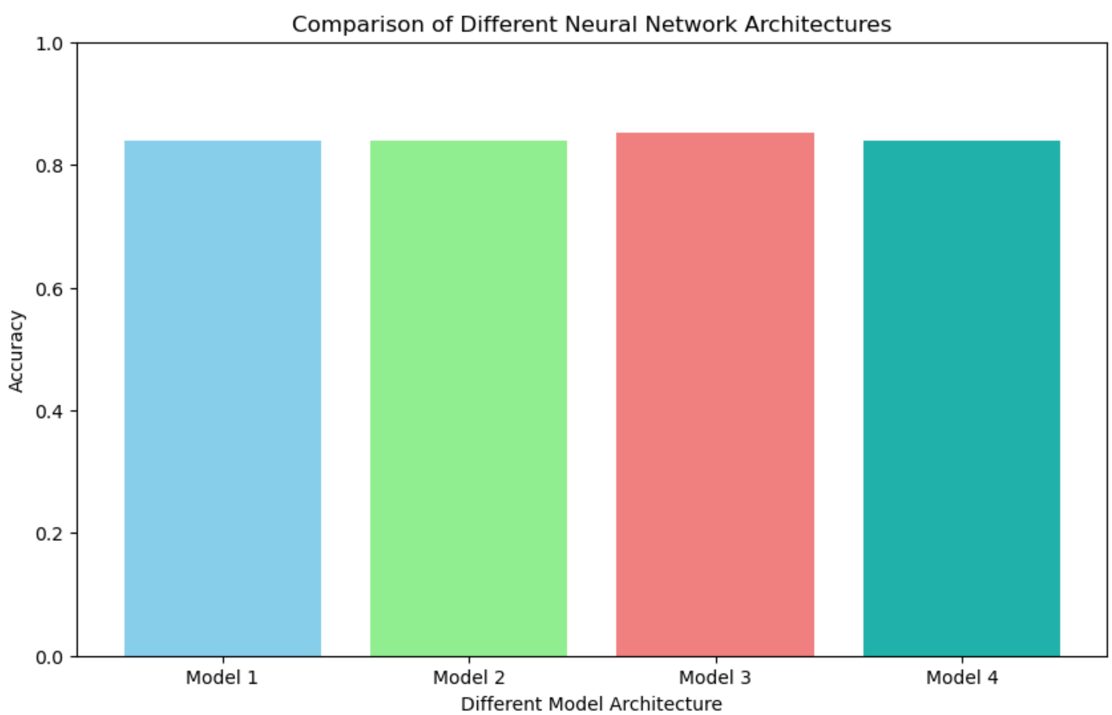
```
    model.compile(optimizer='adam', loss='binary_crossentropy',␣
↪metrics=['accuracy'])
    print(model.layers)
    history = model.fit(X_train, y_train, epochs=10, batch_size=32,␣
↪validation_split=0.2)
    histories.append(history)

# Extracting accuracies
val_accuracies = []

for history in histories:
    val_accuracy = history.history['val_accuracy'][-1]
    val_accuracies.append(val_accuracy)
# Plotting
plt.figure(figsize=(10, 6))

colors = ['skyblue', 'lightgreen', 'lightcoral', 'lightseagreen']

for i, acc in enumerate(val_accuracies):
    plt.bar(i, acc, color=colors[i])

plt.xticks(np.arange(len(models)), ['Model 1', 'Model 2', 'Model 3', 'Model 4'])
plt.xlabel('Different Model Architecture')
plt.ylabel('Accuracy')
plt.title('Comparison of Different Neural Network Architectures')
plt.ylim([0, 1])
plt.show()
```

```
[<keras.layers.core.dense.Dense object at 0x756c2c325760>,
<keras.layers.core.dense.Dense object at 0x756c2f638fd0>,
<keras.layers.core.dense.Dense object at 0x756b8537d610>]
Epoch 1/10
18/18 [==============================] - 0s 5ms/step - loss: 0.6629 - accuracy:
0.6257 - val_loss: 0.6087 - val_accuracy: 0.7622
Epoch 2/10
18/18 [==============================] - 0s 1ms/step - loss: 0.5964 - accuracy:
0.7487 - val_loss: 0.5463 - val_accuracy: 0.8182
Epoch 3/10
18/18 [==============================] - 0s 2ms/step - loss: 0.5534 - accuracy:
0.7698 - val_loss: 0.5073 - val_accuracy: 0.8112
Epoch 4/10
18/18 [==============================] - 0s 1ms/step - loss: 0.5250 - accuracy:
0.7733 - val_loss: 0.4784 - val_accuracy: 0.8112
Epoch 5/10
18/18 [==============================] - 0s 1ms/step - loss: 0.5014 - accuracy:
0.7592 - val_loss: 0.4515 - val_accuracy: 0.8252
Epoch 6/10
```

```
18/18 [==============================] - 0s 1ms/step - loss: 0.4843 - accuracy:
0.7698 - val_loss: 0.4322 - val_accuracy: 0.8322
Epoch 7/10
18/18 [==============================] - 0s 1ms/step - loss: 0.4714 - accuracy:
0.7856 - val_loss: 0.4206 - val_accuracy: 0.8392
Epoch 8/10
18/18 [==============================] - 0s 1ms/step - loss: 0.4651 - accuracy:
0.7909 - val_loss: 0.4157 - val_accuracy: 0.8252
Epoch 9/10
18/18 [==============================] - 0s 1ms/step - loss: 0.4619 - accuracy:
0.7891 - val_loss: 0.4087 - val_accuracy: 0.8322
Epoch 10/10
18/18 [==============================] - 0s 1ms/step - loss: 0.4557 - accuracy:
0.7961 - val_loss: 0.4009 - val_accuracy: 0.8392
[<keras.layers.core.dense.Dense object at 0x756b8537da30>,
<keras.layers.core.dense.Dense object at 0x756b8537d9a0>,
<keras.layers.core.dense.Dense object at 0x756b8536b970>,
<keras.layers.core.dense.Dense object at 0x756b8536b4c0>]
Epoch 1/10
18/18 [==============================] - 0s 5ms/step - loss: 0.6111 - accuracy:
0.6643 - val_loss: 0.4966 - val_accuracy: 0.8042
Epoch 2/10
18/18 [==============================] - 0s 1ms/step - loss: 0.5031 - accuracy:
0.7821 - val_loss: 0.4222 - val_accuracy: 0.8252
Epoch 3/10
18/18 [==============================] - 0s 1ms/step - loss: 0.4669 - accuracy:
0.7873 - val_loss: 0.3962 - val_accuracy: 0.8392
Epoch 4/10
18/18 [==============================] - 0s 2ms/step - loss: 0.4513 - accuracy:
0.7996 - val_loss: 0.3954 - val_accuracy: 0.8322
Epoch 5/10
18/18 [==============================] - 0s 2ms/step - loss: 0.4374 - accuracy:
0.7926 - val_loss: 0.3818 - val_accuracy: 0.8392
Epoch 6/10
18/18 [==============================] - 0s 2ms/step - loss: 0.4340 - accuracy:
0.7996 - val_loss: 0.3791 - val_accuracy: 0.8462
Epoch 7/10
18/18 [==============================] - 0s 1ms/step - loss: 0.4293 - accuracy:
0.8137 - val_loss: 0.3963 - val_accuracy: 0.8252
Epoch 8/10
18/18 [==============================] - 0s 2ms/step - loss: 0.4195 - accuracy:
0.8172 - val_loss: 0.3759 - val_accuracy: 0.8392
Epoch 9/10
18/18 [==============================] - 0s 2ms/step - loss: 0.4159 - accuracy:
0.8207 - val_loss: 0.3801 - val_accuracy: 0.8322
Epoch 10/10
18/18 [==============================] - 0s 2ms/step - loss: 0.4132 - accuracy:
0.8260 - val_loss: 0.3746 - val_accuracy: 0.8392
```

```
[<keras.layers.core.dense.Dense object at 0x756b853691c0>,
<keras.layers.core.dense.Dense object at 0x756b853755b0>,
<keras.layers.core.dense.Dense object at 0x756b853029a0>,
<keras.layers.core.dense.Dense object at 0x756b85302ca0>,
<keras.layers.core.dense.Dense object at 0x756b85308070>,
<keras.layers.core.dense.Dense object at 0x756b853082e0>,
<keras.layers.core.dense.Dense object at 0x756b853085e0>,
<keras.layers.core.dense.Dense object at 0x756b853088e0>,
<keras.layers.core.dense.Dense object at 0x756b853022e0>,
<keras.layers.core.dense.Dense object at 0x756b852fe610>]
Epoch 1/10
18/18 [==============================] - 0s 7ms/step - loss: 0.6510 - accuracy:
0.6169 - val_loss: 0.5618 - val_accuracy: 0.6503
Epoch 2/10
18/18 [==============================] - 0s 2ms/step - loss: 0.5535 - accuracy:
0.7118 - val_loss: 0.4957 - val_accuracy: 0.8042
Epoch 3/10
18/18 [==============================] - 0s 2ms/step - loss: 0.5058 - accuracy:
0.7733 - val_loss: 0.4480 - val_accuracy: 0.8252
Epoch 4/10
18/18 [==============================] - 0s 2ms/step - loss: 0.4897 - accuracy:
0.7821 - val_loss: 0.4279 - val_accuracy: 0.8322
Epoch 5/10
18/18 [==============================] - 0s 2ms/step - loss: 0.4764 - accuracy:
0.7873 - val_loss: 0.4246 - val_accuracy: 0.8252
Epoch 6/10
18/18 [==============================] - 0s 2ms/step - loss: 0.4596 - accuracy:
0.7821 - val_loss: 0.4138 - val_accuracy: 0.8462
Epoch 7/10
18/18 [==============================] - 0s 2ms/step - loss: 0.4461 - accuracy:
0.7891 - val_loss: 0.3926 - val_accuracy: 0.8531
Epoch 8/10
18/18 [==============================] - 0s 2ms/step - loss: 0.4272 - accuracy:
0.7996 - val_loss: 0.3886 - val_accuracy: 0.8531
Epoch 9/10
18/18 [==============================] - 0s 2ms/step - loss: 0.4233 - accuracy:
0.8014 - val_loss: 0.3962 - val_accuracy: 0.8531
Epoch 10/10
18/18 [==============================] - 0s 2ms/step - loss: 0.4157 - accuracy:
0.8172 - val_loss: 0.3829 - val_accuracy: 0.8531
[<keras.layers.core.dense.Dense object at 0x756b852fe9a0>,
<keras.layers.core.dense.Dense object at 0x756b852fe400>]
Epoch 1/10
18/18 [==============================] - 0s 5ms/step - loss: 0.6606 - accuracy:
0.6450 - val_loss: 0.6075 - val_accuracy: 0.8042
Epoch 2/10
18/18 [==============================] - 0s 1ms/step - loss: 0.5964 - accuracy:
0.7610 - val_loss: 0.5477 - val_accuracy: 0.8042
```

```
Epoch 3/10
18/18 [==============================] - 0s 1ms/step - loss: 0.5521 - accuracy:
0.7592 - val_loss: 0.5064 - val_accuracy: 0.8112
Epoch 4/10
18/18 [==============================] - 0s 1ms/step - loss: 0.5231 - accuracy:
0.7680 - val_loss: 0.4753 - val_accuracy: 0.7902
Epoch 5/10
18/18 [==============================] - 0s 1ms/step - loss: 0.5024 - accuracy:
0.7680 - val_loss: 0.4529 - val_accuracy: 0.8112
Epoch 6/10
18/18 [==============================] - 0s 1ms/step - loss: 0.4876 - accuracy:
0.7627 - val_loss: 0.4378 - val_accuracy: 0.8252
Epoch 7/10
18/18 [==============================] - 0s 1ms/step - loss: 0.4768 - accuracy:
0.7733 - val_loss: 0.4267 - val_accuracy: 0.8322
Epoch 8/10
18/18 [==============================] - 0s 1ms/step - loss: 0.4715 - accuracy:
0.7838 - val_loss: 0.4211 - val_accuracy: 0.8392
Epoch 9/10
18/18 [==============================] - 0s 2ms/step - loss: 0.4646 - accuracy:
0.7856 - val_loss: 0.4136 - val_accuracy: 0.8392
Epoch 10/10
18/18 [==============================] - 0s 2ms/step - loss: 0.4606 - accuracy:
0.7768 - val_loss: 0.4104 - val_accuracy: 0.8392
```



0.3.11 WITH SKLEARN

```
[ ]: models = [
         MLPClassifier(hidden_layer_sizes=(32,), activation='relu', max_iter=100),
         MLPClassifier(hidden_layer_sizes=(64, 100, 100), activation='relu',␣
      ↪max_iter=100),
         MLPClassifier(hidden_layer_sizes=(10, 20, 30, 40, 50, 60, 70, 80, 90),␣
      ↪activation='relu', max_iter=100),
         MLPClassifier(hidden_layer_sizes=(100,), activation='relu', max_iter=100)
     ]
     plt.figure(figsize=(10, 6))

     # Define colors for bars
     colors = ['skyblue', 'lightgreen', 'lightcoral', 'lightseagreen']

     for i, history in enumerate(histories):
         val_accuracy = history.history['val_accuracy'][-1]
         plt.bar(i, val_accuracy, color=colors[i])

     plt.xticks(np.arange(len(models)), ['Model 1', 'Model 2', 'Model 3', 'Model 4'])
     plt.xlabel('Different Model Architecture')
     plt.ylabel('Accuracy')
     plt.title('Comparison of Different Neural Network Architectures')
     plt.ylim([0, 1])
     plt.show()
```