Deep Learning : Assignment 03

Name: Jatan Sahu

ID: 202218061

```
# standard libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler,LabelEncoder
from sklearn.model_selection import train_test_split
import seaborn as sb
sb.set_style("white")
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler,PolynomialFeatures
from sklearn.impute import KNNImputer
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import mean_squared_error,mean_absolute_error,r2_score
import missingno as msno
import warnings
warnings.filterwarnings('ignore')
#required keras libraries
import tensorflow as tf
from tensorflow.keras.models import Sequential,Model
from tensorflow.keras.layers import Dense,Input,Dropout
from tensorflow.keras.utils import to_categorical,plot_model
```

Task 1: Regression

```
house = pd.read_csv("https://raw.githubusercontent.com/Jatansahu/DEEP_LEARNING_ASSIGNMENTS/main/LAB_02
```

```
house.info()
```

```
 25  MasVnrType     1452 non-null   object
 26  MasVnrArea     1452 non-null   float64
 27  ExterQual      1460 non-null   object
 28  ExterCond      1460 non-null   object
 29  Foundation     1460 non-null   object
 30  BsmtQual       1423 non-null   object
 31  BsmtCond       1423 non-null   object
 32  BsmtExposure   1422 non-null   object
```

```
 63  GarageQual     1379 non-null   object
 64  GarageCond     1379 non-null   object
 65  PavedDrive     1460 non-null   object
 66  WoodDeckSF     1460 non-null   int64
 67  OpenPorchSF    1460 non-null   int64
 68  EnclosedPorch  1460 non-null   int64
 69  3SsnPorch      1460 non-null   int64
 70  ScreenPorch    1460 non-null   int64
 71  PoolArea       1460 non-null   int64
 72  PoolQC         7 non-null      object
 73  Fence          281 non-null    object
 74  MiscFeature    54 non-null     object
 75  MiscVal        1460 non-null   int64
 76  MoSold         1460 non-null   int64
 77  YrSold         1460 non-null   int64
 78  SaleType       1460 non-null   object
 79  SaleCondition  1460 non-null   object
 80  SalePrice      1460 non-null   int64
dtypes: float64(3), int64(35), object(43)
memory usage: 924.0+ KB
```

```
for i in house.columns:
    if house[i].isnull().sum()>0:
        print(i, '\t', house[i].isnull().sum())
```

```
LotFrontage     259
Alley   1369
MasVnrType      8
MasVnrArea      8
BsmtQual        37
BsmtCond        37
BsmtExposure    38
BsmtFinType1    37
BsmtFinType2    38
Electrical      1
FireplaceQu     690
GarageType      81
GarageYrBlt     81
GarageFinish    81
GarageQual      81
GarageCond      81
PoolQC  1453
Fence   1179
MiscFeature     1406
```

```
house.drop(['Id','Alley','FireplaceQu','PoolQC','Fence','MiscFeature'], axis = 1, inplace = True)
```

```
house['LotFrontage'].fillna(house['LotFrontage'].mean(),inplace=True)
house['MasVnrArea'].fillna(house['MasVnrArea'].mean(), inplace=True)
house['GarageYrBlt'].fillna(house['GarageYrBlt'].mode(), inplace=True)
```

```
m_col=[]
for i in house.columns:
    if house[i].isnull().sum()>0:
        m_col.append(i)

for i in m_col:
    house[i].fillna(house[i].mode(), inplace=True)
```

```
cat_cols = house.select_dtypes(exclude='number').columns.to_list()
num_cols = house.select_dtypes(include='number').columns.to_list()
```

```
encoder = LabelEncoder()
for col in cat_cols:
    house[col] = encoder.fit_transform(house[col])
```

```
for col in num_cols:
    sc = StandardScaler()
    house[col] = sc.fit_transform(house[col].values.reshape(-1,1))
```

```
corr_mat = house.corr()
corr_mat
```

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandC
MSSubClass	1.000000	0.035900	-0.357056	-0.139781	-0.024969	0.119289	-0.
MSZoning	0.035900	1.000000	-0.106363	-0.034452	0.087654	0.061887	-0.
LotFrontage	-0.357056	-0.106363	1.000000	0.306795	-0.037323	-0.144931	-0.
LotArea	-0.139781	-0.034452	0.306795	1.000000	-0.197131	-0.165315	-0.
Street	-0.024969	0.087654	-0.037323	-0.197131	1.000000	-0.010224	0
...	...	...	...	...	...	...	
MoSold	-0.013585	-0.031496	0.010158	0.001205	0.003690	-0.033455	-0
YrSold	-0.021407	-0.020628	0.006768	-0.014261	-0.025043	0.036449	0.
SaleType	0.012464	0.097437	-0.030846	0.012292	0.014339	-0.000911	-0.
SaleCondition	-0.024940	0.009494	0.058464	0.034169	0.006064	-0.038118	0.
SalePrice	-0.084284	-0.166872	0.334901	0.263843	0.041036	-0.255580	0.

```
cols_x = ((corr_mat['SalePrice'] > abs(0.615))[:-1])
cols_x = cols_x[cols_x==True].index
cols_x
```

```
Index(['OverallQual', 'GrLivArea', 'GarageCars', 'GarageArea'], dtype='object')
```

```
x1 = house[cols_x]
y1 = house['SalePrice']
```

```
def Model_Evaluation(x, y, model):
    x_train,x_part,y_train,y_part = train_test_split(x,y,test_size = 0.25,random_state = 67)
    x_test,x_val,y_test,y_val = train_test_split(x_part,y_part,test_size = 0.4,random_state = 67)

    model.fit(x_train,y_train)
    y_pt = model.predict(x_train)
    y_pred = model.predict(x_val)
    print("Model Performance on Validation Data:")
    print("R2 Score:", r2_score(y_val,y_pred))
    print("Mean Square Error:", mean_squared_error(y_val,y_pred))
    print("Mean Absolute Error", mean_absolute_error(y_val,y_pred))
    tr= model.score(x_train,y_train)
    va= model.score(x_val, y_val)
    print("\nTraining Accuracy:", tr)
    print("Validation Accuracy:", va)
```

```
reg = LinearRegression()
Model_Evaluation(x1, y1, reg)
```

```
Model Performance on Validation Data:
R2 Score: 0.770332035999096
Mean Square Error: 0.16372485526381075
Mean Absolute Error 0.31814547253539976

Training Accuracy: 0.7363611940603583
Validation Accuracy: 0.770332035999096
```

```
poly2 = PolynomialFeatures(degree = 2)
x2 = poly2.fit_transform(x1)

poly3 = PolynomialFeatures(degree = 3)
x3 = poly3.fit_transform(x1)

poly5 = PolynomialFeatures(degree = 5)
x5 = poly5.fit_transform(x1)
```

```
reg2 = LinearRegression()
Model_Evaluation(x2, y1, reg2)
```

```
Model Performance on Validation Data:
R2 Score: 0.8062060207654109
Mean Square Error: 0.13815114066607986
Mean Absolute Error 0.2773622998247046

Training Accuracy: 0.8150231368729806
Validation Accuracy: 0.8062060207654109
```

```
reg3 = LinearRegression()
Model_Evaluation(x3, y1, reg3)
```

```
Model Performance on Validation Data:
R2 Score: 0.7917026723248876
Mean Square Error: 0.1484902344730679
Mean Absolute Error 0.27647964407257253

Training Accuracy: 0.8292855708706803
Validation Accuracy: 0.7917026723248876
```

```
def Model_Testing(x, y, model):
    x_train,x_part,y_train,y_part = train_test_split(x,y,test_size = 0.25,random_state = 67)
    x_test,x_val,y_test,y_val = train_test_split(x_part,y_part,test_size = 0.4,random_state = 67)

    model.fit(x_train,y_train)
    y_pred = model.predict(x_test)
    print("Model Performance on Test Data:")
    print("R2 Score:", r2_score(y_test,y_pred))
    print("Mean Square Error:", mean_squared_error(y_test,y_pred))
    print("Mean Absolute Error", mean_absolute_error(y_test,y_pred))
    te= model.score(x_test, y_test)
    print("\nTesting Accuracy:", te)
```

```
Model_Testing(x3, y1, reg3)
```

```
Model Performance on Test Data:
R2 Score: 0.8065649082770836
Mean Square Error: 0.15772875315052456
Mean Absolute Error 0.29441183196461623

Testing Accuracy: 0.8065649082770836
```

Regression using ANN

```
x_train,x_part,y_train,y_part = train_test_split(x1,y1,test_size = 0.25,random_state = 67)
x_test,x_val,y_test,y_val = train_test_split(x_part,y_part,test_size = 0.4,random_state = 67)
```

```
def plot_function(history):
    plt.figure(figsize=(15,5))
    plt.plot(history.history['loss'],color = 'red',label = 'train_loss')
    plt.title('Loss and val_loss')
    plt.plot(history.history['val_loss'],color = 'green',label = 'val_loss')
    plt.legend()
```

```
x_train.shape
```

```
(1095, 4)
```

Model 1

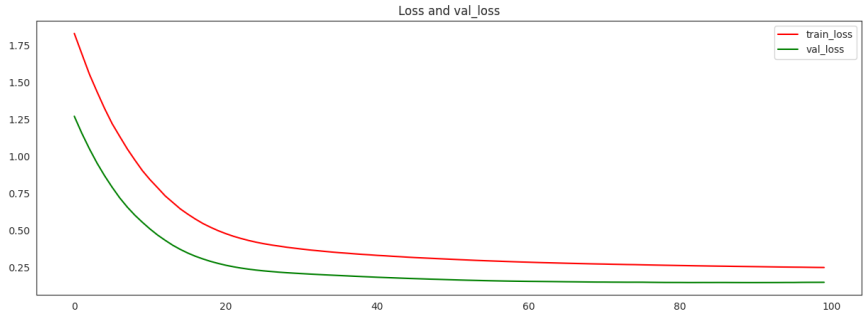
```
model1 = Sequential()
model1.add(Dense(8, input_shape=(4,), activation='relu'))
model1.add(Dense(1, activation=None))
model1.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 8)                 40

 dense_1 (Dense)             (None, 1)                 9

=================================================================
Total params: 49
Trainable params: 49
Non-trainable params: 0
_________________________________________________________________
```

```
model1.compile(loss = tf.keras.losses.MeanSquaredError() ,optimizer = 'adam',metrics = ['mse'])
history1 = model1.fit(x_train,y_train,epochs = 100,batch_size = 200,validation_data = (x_val,y_val))
```

```
Epoch 72/100
6/6 [==============================] - 0s 8ms/step - loss: 0.2713 - mse: 0.2713 - val_loss: 0.1506 - val_mse: 0.1506
Epoch 73/100
6/6 [==============================] - 0s 21ms/step - loss: 0.2704 - mse: 0.2704 - val_loss: 0.1504 - val_mse: 0.1504
Epoch 74/100
6/6 [==============================] - 0s 19ms/step - loss: 0.2694 - mse: 0.2694 - val_loss: 0.1502 - val_mse: 0.1502
Epoch 75/100
6/6 [==============================] - 0s 14ms/step - loss: 0.2687 - mse: 0.2687 - val_loss: 0.1502 - val_mse: 0.1502
Epoch 76/100
6/6 [==============================] - 0s 16ms/step - loss: 0.2676 - mse: 0.2676 - val_loss: 0.1501 - val_mse: 0.1501
Epoch 77/100
6/6 [==============================] - 0s 16ms/step - loss: 0.2666 - mse: 0.2666 - val_loss: 0.1495 - val_mse: 0.1495
Epoch 78/100
6/6 [==============================] - 0s 12ms/step - loss: 0.2656 - mse: 0.2656 - val_loss: 0.1490 - val_mse: 0.1490
Epoch 79/100
6/6 [==============================] - 0s 14ms/step - loss: 0.2647 - mse: 0.2647 - val_loss: 0.1486 - val_mse: 0.1486
Epoch 80/100
6/6 [==============================] - 0s 22ms/step - loss: 0.2638 - mse: 0.2638 - val_loss: 0.1485 - val_mse: 0.1485
Epoch 81/100
6/6 [==============================] - 0s 23ms/step - loss: 0.2629 - mse: 0.2629 - val_loss: 0.1484 - val_mse: 0.1484
Epoch 82/100
6/6 [==============================] - 0s 17ms/step - loss: 0.2620 - mse: 0.2620 - val_loss: 0.1481 - val_mse: 0.1481
Epoch 83/100
6/6 [==============================] - 0s 16ms/step - loss: 0.2611 - mse: 0.2611 - val_loss: 0.1481 - val_mse: 0.1481
Epoch 84/100
6/6 [==============================] - 0s 15ms/step - loss: 0.2603 - mse: 0.2603 - val_loss: 0.1483 - val_mse: 0.1483
Epoch 85/100
6/6 [==============================] - 0s 16ms/step - loss: 0.2594 - mse: 0.2594 - val_loss: 0.1482 - val_mse: 0.1482
Epoch 86/100
6/6 [==============================] - 0s 17ms/step - loss: 0.2587 - mse: 0.2587 - val_loss: 0.1484 - val_mse: 0.1484
Epoch 87/100
6/6 [==============================] - 0s 17ms/step - loss: 0.2580 - mse: 0.2580 - val_loss: 0.1483 - val_mse: 0.1483
Epoch 88/100
6/6 [==============================] - 0s 20ms/step - loss: 0.2572 - mse: 0.2572 - val_loss: 0.1481 - val_mse: 0.1481
Epoch 89/100
6/6 [==============================] - 0s 19ms/step - loss: 0.2564 - mse: 0.2564 - val_loss: 0.1479 - val_mse: 0.1479
Epoch 90/100
6/6 [==============================] - 0s 24ms/step - loss: 0.2558 - mse: 0.2558 - val_loss: 0.1478 - val_mse: 0.1478
Epoch 91/100
6/6 [==============================] - 0s 20ms/step - loss: 0.2551 - mse: 0.2551 - val_loss: 0.1478 - val_mse: 0.1478
Epoch 92/100
6/6 [==============================] - 0s 13ms/step - loss: 0.2544 - mse: 0.2544 - val_loss: 0.1479 - val_mse: 0.1479
Epoch 93/100
6/6 [==============================] - 0s 12ms/step - loss: 0.2537 - mse: 0.2537 - val_loss: 0.1480 - val_mse: 0.1480
Epoch 94/100
6/6 [==============================] - 0s 15ms/step - loss: 0.2530 - mse: 0.2530 - val_loss: 0.1481 - val_mse: 0.1481
Epoch 95/100
6/6 [==============================] - 0s 25ms/step - loss: 0.2524 - mse: 0.2524 - val_loss: 0.1485 - val_mse: 0.1485
Epoch 96/100
6/6 [==============================] - 0s 21ms/step - loss: 0.2519 - mse: 0.2519 - val_loss: 0.1484 - val_mse: 0.1484
Epoch 97/100
6/6 [==============================] - 0s 16ms/step - loss: 0.2515 - mse: 0.2515 - val_loss: 0.1491 - val_mse: 0.1491
Epoch 98/100
6/6 [==============================] - 0s 17ms/step - loss: 0.2508 - mse: 0.2508 - val_loss: 0.1494 - val_mse: 0.1494
Epoch 99/100
6/6 [==============================] - 0s 16ms/step - loss: 0.2501 - mse: 0.2501 - val_loss: 0.1495 - val_mse: 0.1495
Epoch 100/100
6/6 [==============================] - 0s 14ms/step - loss: 0.2497 - mse: 0.2497 - val_loss: 0.1496 - val_mse: 0.1496
```

```
plot_function(history1)
```



Model 2

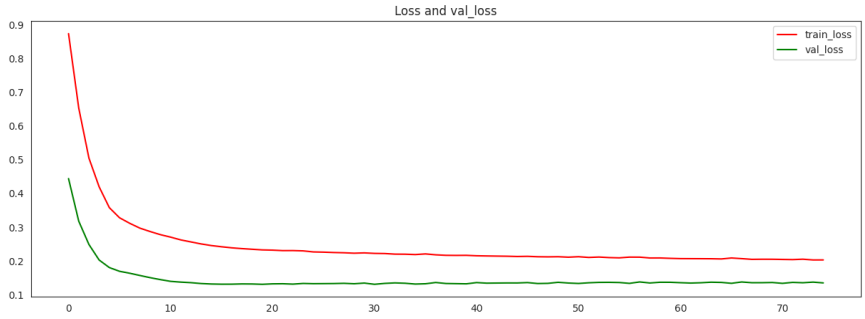
```
model2 = Sequential()
model2.add(Dense(32, input_shape=(4,), activation='relu'))
model2.add(Dense(1, activation=None))
model2.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_2 (Dense)             (None, 32)                160

 dense_3 (Dense)             (None, 1)                 33

=================================================================
Total params: 193
Trainable params: 193
Non-trainable params: 0
_________________________________________________________________
```

```
model2.compile(loss = tf.keras.losses.MeanSquaredError() ,optimizer = 'adam',metrics = ['mse'])
history2 = model2.fit(x_train,y_train,epochs = 75,batch_size = 80,validation_data = (x_val,y_val))
```

```
14/14 [==============================] - 0s 12ms/step - loss: 0.2068 - mse: 0.2068 - val_loss: 0.1344 - val_mse: 0.1344
Epoch 63/75
14/14 [==============================] - 0s 15ms/step - loss: 0.2067 - mse: 0.2067 - val_loss: 0.1353 - val_mse: 0.1353
Epoch 64/75
14/14 [==============================] - 0s 17ms/step - loss: 0.2065 - mse: 0.2065 - val_loss: 0.1370 - val_mse: 0.1370
Epoch 65/75
14/14 [==============================] - 0s 19ms/step - loss: 0.2059 - mse: 0.2059 - val_loss: 0.1363 - val_mse: 0.1363
Epoch 66/75
14/14 [==============================] - 0s 21ms/step - loss: 0.2088 - mse: 0.2088 - val_loss: 0.1337 - val_mse: 0.1337
Epoch 67/75
14/14 [==============================] - 0s 15ms/step - loss: 0.2069 - mse: 0.2069 - val_loss: 0.1376 - val_mse: 0.1376
Epoch 68/75
14/14 [==============================] - 0s 14ms/step - loss: 0.2047 - mse: 0.2047 - val_loss: 0.1354 - val_mse: 0.1354
Epoch 69/75
14/14 [==============================] - 0s 15ms/step - loss: 0.2050 - mse: 0.2050 - val_loss: 0.1355 - val_mse: 0.1355
Epoch 70/75
14/14 [==============================] - 0s 19ms/step - loss: 0.2050 - mse: 0.2050 - val_loss: 0.1360 - val_mse: 0.1360
Epoch 71/75
14/14 [==============================] - 0s 21ms/step - loss: 0.2045 - mse: 0.2045 - val_loss: 0.1336 - val_mse: 0.1336
Epoch 72/75
14/14 [==============================] - 0s 24ms/step - loss: 0.2040 - mse: 0.2040 - val_loss: 0.1364 - val_mse: 0.1364
Epoch 73/75
14/14 [==============================] - 0s 25ms/step - loss: 0.2051 - mse: 0.2051 - val_loss: 0.1353 - val_mse: 0.1353
Epoch 74/75
14/14 [==============================] - 0s 23ms/step - loss: 0.2029 - mse: 0.2029 - val_loss: 0.1372 - val_mse: 0.1372
Epoch 75/75
14/14 [==============================] - 0s 31ms/step - loss: 0.2030 - mse: 0.2030 - val_loss: 0.1348 - val_mse: 0.1348
```

```
plot_function(history2)
```



Model 3

```
model3 = Sequential()
model3.add(Dense(8, input_shape=(4,), activation='relu'))
model3.add(Dense(16, activation='relu'))
model3.add(Dense(32, activation='relu'))
model3.add(Dense(16, activation='relu'))
model3.add(Dense(8, activation='relu'))
model3.add(Dense(1, activation=None))
model3.summary()
```

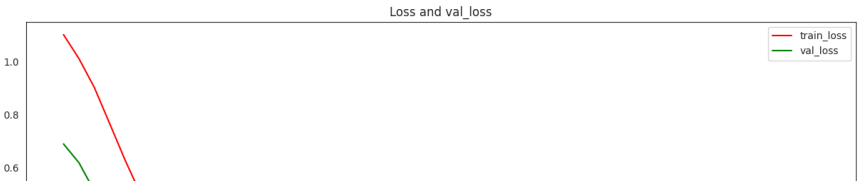
```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_4 (Dense)             (None, 8)                 40

 dense_5 (Dense)             (None, 16)                144

 dense_6 (Dense)             (None, 32)                544

 dense_7 (Dense)             (None, 16)                528

 dense_8 (Dense)             (None, 8)                 136

 dense_9 (Dense)             (None, 1)                 9

=================================================================
```

```
Total params: 1,401
Trainable params: 1,401
Non-trainable params: 0
_________________________________________________________________
```

```
model3.compile(loss = tf.keras.losses.MeanSquaredError() ,optimizer = 'adam',metrics = ['mse'])
history3 = model3.fit(x_train,y_train,epochs = 50,batch_size = 128,validation_data = (x_val,y_val))
```

```
Epoch 22/50
9/9 [==============================] - 0s 26ms/step - loss: 0.2484 - mse: 0.2484 - val_loss: 0.1660 - val_mse: 0.1660
Epoch 23/50
9/9 [==============================] - 0s 36ms/step - loss: 0.2486 - mse: 0.2486 - val_loss: 0.1638 - val_mse: 0.1638
Epoch 24/50
9/9 [==============================] - 0s 30ms/step - loss: 0.2459 - mse: 0.2459 - val_loss: 0.1698 - val_mse: 0.1698
Epoch 25/50
9/9 [==============================] - 0s 25ms/step - loss: 0.2443 - mse: 0.2443 - val_loss: 0.1643 - val_mse: 0.1643
Epoch 26/50
9/9 [==============================] - 0s 12ms/step - loss: 0.2440 - mse: 0.2440 - val_loss: 0.1633 - val_mse: 0.1633
Epoch 27/50
9/9 [==============================] - 0s 17ms/step - loss: 0.2424 - mse: 0.2424 - val_loss: 0.1663 - val_mse: 0.1663
Epoch 28/50
9/9 [==============================] - 0s 21ms/step - loss: 0.2413 - mse: 0.2413 - val_loss: 0.1603 - val_mse: 0.1603
Epoch 29/50
9/9 [==============================] - 0s 13ms/step - loss: 0.2400 - mse: 0.2400 - val_loss: 0.1667 - val_mse: 0.1667
Epoch 30/50
9/9 [==============================] - 0s 15ms/step - loss: 0.2387 - mse: 0.2387 - val_loss: 0.1624 - val_mse: 0.1624
Epoch 31/50
9/9 [==============================] - 0s 37ms/step - loss: 0.2358 - mse: 0.2358 - val_loss: 0.1622 - val_mse: 0.1622
Epoch 32/50
9/9 [==============================] - 0s 32ms/step - loss: 0.2355 - mse: 0.2355 - val_loss: 0.1640 - val_mse: 0.1640
Epoch 33/50
9/9 [==============================] - 0s 36ms/step - loss: 0.2337 - mse: 0.2337 - val_loss: 0.1606 - val_mse: 0.1606
Epoch 34/50
9/9 [==============================] - 0s 19ms/step - loss: 0.2337 - mse: 0.2337 - val_loss: 0.1651 - val_mse: 0.1651
Epoch 35/50
9/9 [==============================] - 0s 13ms/step - loss: 0.2324 - mse: 0.2324 - val_loss: 0.1643 - val_mse: 0.1643
Epoch 36/50
9/9 [==============================] - 0s 7ms/step - loss: 0.2308 - mse: 0.2308 - val_loss: 0.1629 - val_mse: 0.1629
Epoch 37/50
9/9 [==============================] - 0s 7ms/step - loss: 0.2298 - mse: 0.2298 - val_loss: 0.1613 - val_mse: 0.1613
Epoch 38/50
9/9 [==============================] - 0s 7ms/step - loss: 0.2291 - mse: 0.2291 - val_loss: 0.1637 - val_mse: 0.1637
Epoch 39/50
9/9 [==============================] - 0s 15ms/step - loss: 0.2291 - mse: 0.2291 - val_loss: 0.1610 - val_mse: 0.1610
Epoch 40/50
9/9 [==============================] - 0s 19ms/step - loss: 0.2281 - mse: 0.2281 - val_loss: 0.1632 - val_mse: 0.1632
Epoch 41/50
9/9 [==============================] - 0s 24ms/step - loss: 0.2301 - mse: 0.2301 - val_loss: 0.1632 - val_mse: 0.1632
Epoch 42/50
9/9 [==============================] - 0s 17ms/step - loss: 0.2271 - mse: 0.2271 - val_loss: 0.1579 - val_mse: 0.1579
Epoch 43/50
9/9 [==============================] - 0s 26ms/step - loss: 0.2248 - mse: 0.2248 - val_loss: 0.1624 - val_mse: 0.1624
Epoch 44/50
9/9 [==============================] - 0s 15ms/step - loss: 0.2234 - mse: 0.2234 - val_loss: 0.1601 - val_mse: 0.1601
Epoch 45/50
9/9 [==============================] - 0s 17ms/step - loss: 0.2228 - mse: 0.2228 - val_loss: 0.1603 - val_mse: 0.1603
Epoch 46/50
9/9 [==============================] - 0s 14ms/step - loss: 0.2251 - mse: 0.2251 - val_loss: 0.1626 - val_mse: 0.1626
Epoch 47/50
9/9 [==============================] - 0s 23ms/step - loss: 0.2225 - mse: 0.2225 - val_loss: 0.1620 - val_mse: 0.1620
Epoch 48/50
9/9 [==============================] - 0s 12ms/step - loss: 0.2262 - mse: 0.2262 - val_loss: 0.1569 - val_mse: 0.1569
Epoch 49/50
9/9 [==============================] - 0s 19ms/step - loss: 0.2212 - mse: 0.2212 - val_loss: 0.1631 - val_mse: 0.1631
Epoch 50/50
9/9 [==============================] - 0s 39ms/step - loss: 0.2182 - mse: 0.2182 - val_loss: 0.1576 - val_mse: 0.1576
```

```
plot_function(history3)
```



Model 1 performs the best, hence checking loss on test data:

```
1 loss1 = model1.evaluate(x_test, y_test, verbose=0)
2 print(f"Mean Squared Error on Test Data: {loss1[0]}")
```

```
Mean Squared Error on Test Data: 0.1773124486207962
```

Task 2: Classification

Classification using ANN

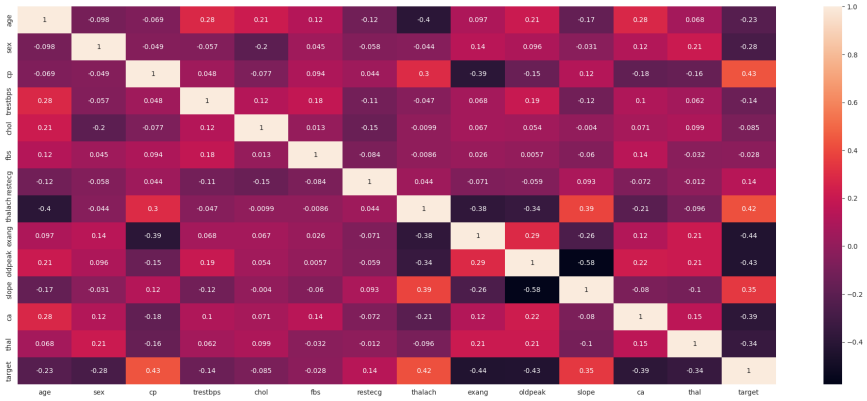
```
1 heart = pd.read_csv("https://raw.githubusercontent.com/Jatansahu/DEEP_LEARNING_ASSIGNMENTS/main/LAB_02
```

```
1 heart.columns
```

```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

```
1 s_col = ['age', 'cp', 'trestbps', 'chol', 'restecg', 'thalach',
2        'oldpeak', 'slope', 'ca', 'thal']
3 for i in s_col:
4     scaler = StandardScaler()
5     heart[i] = scaler.fit_transform(heart[i].values.reshape(-1,1))
```

```
1 plt.figure(figsize = (25,10))
2 sb.heatmap(heart.corr(),annot = True);
```




```
1 x4 = heart.iloc[:,:-1]
2 y4 = heart['target']
```

```
x_train,x_part,y_train,y_part = train_test_split(x4,y4,test_size = 0.25,random_state = 67)
x_test,x_val,y_test,y_val = train_test_split(x_part,y_part,test_size = 0.4,random_state = 67)
```

```
def plot_loss_accuracy(history):
    fig,ax = plt.subplots(1,2,figsize = (25,5))
    ax[0].plot(history.history['loss'],color = 'red',label = 'train_loss')
    ax[0].set_title('Loss and val_loss')
    ax[0].plot(history.history['val_loss'],color = 'green',label = 'val_loss')
    ax[0].legend()
    ax[1].plot(history.history['accuracy'],color = 'orange',label = 'train_accuracy')
    ax[1].set_title('accuracy and val_accuracy')
    ax[1].plot(history.history['val_accuracy'],color = 'black',label = 'val_accuarcy')
    ax[1].legend()
```

```
x_train.shape
```

```
(227, 13)
```

Classification Model 1

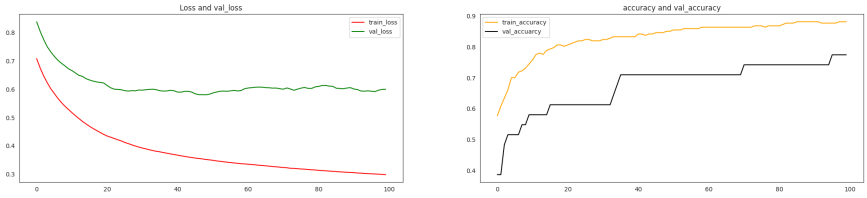
```
model4 = Sequential()
model4.add(Dense(16, input_shape=(13,), activation='relu'))
model4.add(Dense(1, activation='sigmoid'))
model4.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_10 (Dense)            (None, 16)                224

 dense_11 (Dense)            (None, 1)                 17

=================================================================
Total params: 241
Trainable params: 241
Non-trainable params: 0
_________________________________________________________________
```

```
model4.compile(loss = 'binary_crossentropy',optimizer = 'adam',metrics = ['accuracy'])
history4 = model4.fit(x_train,y_train,epochs = 100,batch_size = 32,validation_data = (x_val,y_val))
```

```
8/8 [==============================] - 0s 10ms/step - loss: 0.3060 - accuracy: 0.8811 - val_loss: 0.6056 - val_accuracy: 0.7419
Epoch 91/100
8/8 [==============================] - 0s 11ms/step - loss: 0.3055 - accuracy: 0.8811 - val_loss: 0.6008 - val_accuracy: 0.7419
Epoch 92/100
8/8 [==============================] - 0s 8ms/step - loss: 0.3042 - accuracy: 0.8811 - val_loss: 0.5988 - val_accuracy: 0.7419
Epoch 93/100
8/8 [==============================] - 0s 9ms/step - loss: 0.3036 - accuracy: 0.8767 - val_loss: 0.5937 - val_accuracy: 0.7419
Epoch 94/100
8/8 [==============================] - 0s 9ms/step - loss: 0.3028 - accuracy: 0.8767 - val_loss: 0.5931 - val_accuracy: 0.7419
Epoch 95/100
8/8 [==============================] - 0s 6ms/step - loss: 0.3022 - accuracy: 0.8767 - val_loss: 0.5943 - val_accuracy: 0.7419
Epoch 96/100
8/8 [==============================] - 0s 6ms/step - loss: 0.3012 - accuracy: 0.8767 - val_loss: 0.5930 - val_accuracy: 0.7742
Epoch 97/100
8/8 [==============================] - 0s 6ms/step - loss: 0.3007 - accuracy: 0.8767 - val_loss: 0.5916 - val_accuracy: 0.7742
Epoch 98/100
8/8 [==============================] - 0s 6ms/step - loss: 0.3001 - accuracy: 0.8811 - val_loss: 0.5966 - val_accuracy: 0.7742
Epoch 99/100
8/8 [==============================] - 0s 6ms/step - loss: 0.2994 - accuracy: 0.8811 - val_loss: 0.5993 - val_accuracy: 0.7742
Epoch 100/100
8/8 [==============================] - 0s 8ms/step - loss: 0.2985 - accuracy: 0.8811 - val_loss: 0.5999 - val_accuracy: 0.7742
```

```
plot_loss_accuracy(history4)
```



Classification Model 2

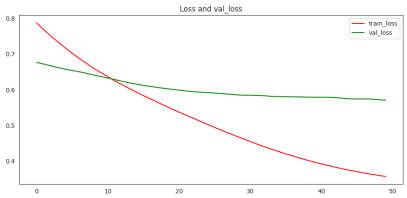
```
model5 = Sequential()
model5.add(Dense(32, input_shape=(13,), activation='relu'))
model5.add(Dense(8, activation='relu'))
model5.add(Dense(1, activation='sigmoid'))
model5.summary()
```

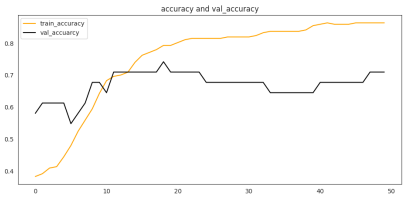
```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_12 (Dense)            (None, 32)                448

 dense_13 (Dense)            (None, 8)                 264

 dense_14 (Dense)            (None, 1)                 9

=================================================================
Total params: 721
Trainable params: 721
Non-trainable params: 0
_________________________________________________________________
```

```
model5.compile(loss = 'binary_crossentropy',optimizer = 'adam',metrics = ['accuracy'])
history5 = model5.fit(x_train,y_train,epochs = 50,batch_size = 100,validation_data = (x_val,y_val))
```

```
3/3 [==============================] - 0s 18ms/step - loss: 0.4701 - accuracy: 0.8194 - val_loss: 0.5857 - val_accuracy: 0.6774
Epoch 30/50
3/3 [==============================] - 0s 28ms/step - loss: 0.4624 - accuracy: 0.8194 - val_loss: 0.5846 - val_accuracy: 0.6774
Epoch 31/50
3/3 [==============================] - 0s 19ms/step - loss: 0.4548 - accuracy: 0.8194 - val_loss: 0.5845 - val_accuracy: 0.6774
Epoch 32/50
3/3 [==============================] - 0s 18ms/step - loss: 0.4474 - accuracy: 0.8238 - val_loss: 0.5841 - val_accuracy: 0.6774
Epoch 33/50
3/3 [==============================] - 0s 19ms/step - loss: 0.4401 - accuracy: 0.8326 - val_loss: 0.5831 - val_accuracy: 0.6774
Epoch 34/50
3/3 [==============================] - 0s 19ms/step - loss: 0.4333 - accuracy: 0.8370 - val_loss: 0.5813 - val_accuracy: 0.6452
Epoch 35/50
3/3 [==============================] - 0s 19ms/step - loss: 0.4268 - accuracy: 0.8370 - val_loss: 0.5807 - val_accuracy: 0.6452
Epoch 36/50
3/3 [==============================] - 0s 17ms/step - loss: 0.4202 - accuracy: 0.8370 - val_loss: 0.5803 - val_accuracy: 0.6452
Epoch 37/50
3/3 [==============================] - 0s 17ms/step - loss: 0.4143 - accuracy: 0.8370 - val_loss: 0.5799 - val_accuracy: 0.6452
Epoch 38/50
3/3 [==============================] - 0s 17ms/step - loss: 0.4082 - accuracy: 0.8370 - val_loss: 0.5797 - val_accuracy: 0.6452
Epoch 39/50
3/3 [==============================] - 0s 17ms/step - loss: 0.4025 - accuracy: 0.8414 - val_loss: 0.5792 - val_accuracy: 0.6452
Epoch 40/50
3/3 [==============================] - 0s 18ms/step - loss: 0.3972 - accuracy: 0.8546 - val_loss: 0.5789 - val_accuracy: 0.6452
Epoch 41/50
3/3 [==============================] - 0s 18ms/step - loss: 0.3921 - accuracy: 0.8590 - val_loss: 0.5788 - val_accuracy: 0.6774
Epoch 42/50
3/3 [==============================] - 0s 17ms/step - loss: 0.3875 - accuracy: 0.8634 - val_loss: 0.5787 - val_accuracy: 0.6774
Epoch 43/50
3/3 [==============================] - 0s 27ms/step - loss: 0.3827 - accuracy: 0.8590 - val_loss: 0.5778 - val_accuracy: 0.6774
Epoch 44/50
3/3 [==============================] - 0s 19ms/step - loss: 0.3783 - accuracy: 0.8590 - val_loss: 0.5759 - val_accuracy: 0.6774
Epoch 45/50
3/3 [==============================] - 0s 17ms/step - loss: 0.3742 - accuracy: 0.8590 - val_loss: 0.5744 - val_accuracy: 0.6774
Epoch 46/50
3/3 [==============================] - 0s 18ms/step - loss: 0.3702 - accuracy: 0.8634 - val_loss: 0.5740 - val_accuracy: 0.6774
Epoch 47/50
3/3 [==============================] - 0s 18ms/step - loss: 0.3666 - accuracy: 0.8634 - val_loss: 0.5744 - val_accuracy: 0.6774
Epoch 48/50
3/3 [==============================] - 0s 26ms/step - loss: 0.3632 - accuracy: 0.8634 - val_loss: 0.5738 - val_accuracy: 0.7097
Epoch 49/50
3/3 [==============================] - 0s 17ms/step - loss: 0.3601 - accuracy: 0.8634 - val_loss: 0.5722 - val_accuracy: 0.7097
Epoch 50/50
3/3 [==============================] - 0s 18ms/step - loss: 0.3569 - accuracy: 0.8634 - val_loss: 0.5703 - val_accuracy: 0.7097
```

```
plot_loss_accuracy(history5)
```





Classification Model 3

```
model6 = Sequential()
model6.add(Dense(8, input_shape=(13,), activation='relu'))
model6.add(Dense(16, activation='relu'))
model6.add(Dense(32, activation='relu'))
model6.add(Dense(8, activation='relu'))
model6.add(Dense(1, activation='sigmoid'))
model6.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_15 (Dense)            (None, 8)                 112

 dense_16 (Dense)            (None, 16)                144

 dense_17 (Dense)            (None, 32)                544

 dense_18 (Dense)            (None, 8)                 264
```

```
 dense_19 (Dense)            (None, 1)                 9

=================================================================
Total params: 1,073
Trainable params: 1,073
Non-trainable params: 0
_________________________________________________________________
```

```
model6.compile(loss = 'binary_crossentropy',optimizer = 'adam',metrics = ['accuracy'])
history6 = model6.fit(x_train,y_train,epochs = 100,batch_size = 32,validation_data = (x_val,y_val))
```

```
Epoch 72/100
8/8 [==============================] - 0s 6ms/step - loss: 0.2517 - accuracy: 0.9031 - val_loss: 0.6000 - val_accuracy: 0.8065
Epoch 73/100
8/8 [==============================] - 0s 6ms/step - loss: 0.2491 - accuracy: 0.9031 - val_loss: 0.6049 - val_accuracy: 0.8065
Epoch 74/100
8/8 [==============================] - 0s 6ms/step - loss: 0.2453 - accuracy: 0.9031 - val_loss: 0.6211 - val_accuracy: 0.8387
Epoch 75/100
8/8 [==============================] - 0s 6ms/step - loss: 0.2438 - accuracy: 0.9075 - val_loss: 0.6252 - val_accuracy: 0.8387
Epoch 76/100
8/8 [==============================] - 0s 6ms/step - loss: 0.2453 - accuracy: 0.8987 - val_loss: 0.5957 - val_accuracy: 0.8065
Epoch 77/100
8/8 [==============================] - 0s 6ms/step - loss: 0.2407 - accuracy: 0.8943 - val_loss: 0.6121 - val_accuracy: 0.8065
Epoch 78/100
8/8 [==============================] - 0s 6ms/step - loss: 0.2378 - accuracy: 0.9031 - val_loss: 0.6392 - val_accuracy: 0.8387
Epoch 79/100
8/8 [==============================] - 0s 7ms/step - loss: 0.2402 - accuracy: 0.9031 - val_loss: 0.6589 - val_accuracy: 0.8387
Epoch 80/100
8/8 [==============================] - 0s 6ms/step - loss: 0.2409 - accuracy: 0.9075 - val_loss: 0.6734 - val_accuracy: 0.8065
Epoch 81/100
8/8 [==============================] - 0s 7ms/step - loss: 0.2360 - accuracy: 0.9031 - val_loss: 0.6597 - val_accuracy: 0.8065
Epoch 82/100
8/8 [==============================] - 0s 6ms/step - loss: 0.2313 - accuracy: 0.8943 - val_loss: 0.6435 - val_accuracy: 0.8065
Epoch 83/100
8/8 [==============================] - 0s 6ms/step - loss: 0.2288 - accuracy: 0.9075 - val_loss: 0.6537 - val_accuracy: 0.8065
Epoch 84/100
8/8 [==============================] - 0s 6ms/step - loss: 0.2260 - accuracy: 0.9075 - val_loss: 0.6631 - val_accuracy: 0.8065
Epoch 85/100
8/8 [==============================] - 0s 9ms/step - loss: 0.2245 - accuracy: 0.9075 - val_loss: 0.6722 - val_accuracy: 0.8065
Epoch 86/100
8/8 [==============================] - 0s 7ms/step - loss: 0.2260 - accuracy: 0.9075 - val_loss: 0.7037 - val_accuracy: 0.8065
Epoch 87/100
8/8 [==============================] - 0s 9ms/step - loss: 0.2226 - accuracy: 0.9119 - val_loss: 0.6871 - val_accuracy: 0.8065
Epoch 88/100
8/8 [==============================] - 0s 8ms/step - loss: 0.2181 - accuracy: 0.9075 - val_loss: 0.6677 - val_accuracy: 0.7742
Epoch 89/100
8/8 [==============================] - 0s 6ms/step - loss: 0.2176 - accuracy: 0.9075 - val_loss: 0.6633 - val_accuracy: 0.7742
Epoch 90/100
8/8 [==============================] - 0s 6ms/step - loss: 0.2139 - accuracy: 0.9119 - val_loss: 0.6882 - val_accuracy: 0.7742
Epoch 91/100
8/8 [==============================] - 0s 6ms/step - loss: 0.2117 - accuracy: 0.9119 - val_loss: 0.6951 - val_accuracy: 0.7742
Epoch 92/100
8/8 [==============================] - 0s 6ms/step - loss: 0.2093 - accuracy: 0.9119 - val_loss: 0.7168 - val_accuracy: 0.8065
Epoch 93/100
8/8 [==============================] - 0s 6ms/step - loss: 0.2078 - accuracy: 0.9295 - val_loss: 0.7666 - val_accuracy: 0.8065
Epoch 94/100
8/8 [==============================] - 0s 9ms/step - loss: 0.2219 - accuracy: 0.9207 - val_loss: 0.7744 - val_accuracy: 0.8065
Epoch 95/100
8/8 [==============================] - 0s 7ms/step - loss: 0.2090 - accuracy: 0.9295 - val_loss: 0.7245 - val_accuracy: 0.7742
Epoch 96/100
8/8 [==============================] - 0s 6ms/step - loss: 0.2054 - accuracy: 0.9207 - val_loss: 0.6766 - val_accuracy: 0.7742
Epoch 97/100
8/8 [==============================] - 0s 6ms/step - loss: 0.2084 - accuracy: 0.9119 - val_loss: 0.6990 - val_accuracy: 0.7742
Epoch 98/100
8/8 [==============================] - 0s 9ms/step - loss: 0.2027 - accuracy: 0.9163 - val_loss: 0.7520 - val_accuracy: 0.8065
Epoch 99/100
8/8 [==============================] - 0s 6ms/step - loss: 0.1991 - accuracy: 0.9163 - val_loss: 0.7597 - val_accuracy: 0.8065
Epoch 100/100
8/8 [==============================] - 0s 9ms/step - loss: 0.1980 - accuracy: 0.9207 - val_loss: 0.7669 - val_accuracy: 0.8065
```

```
plot_loss_accuracy(history6)
```

Classification Model 1 performs the best, hence checking accuracy on test data:

```
1  loss4 = model4.evaluate(x_test, y_test, verbose=0)
2  print(f"Accuracy on test data: {loss4[1]:.2f}")
```

```
Accuracy on test data: 0.82
```

✓ 0s completed at 11:00 PM