Student Placement Prediction Using Deep Learning

Dataset : https://archive.ics.uci.edu/dataset/320/student+performance

By: Praveen Choudhary

In [ ]:
```
# !pip install ucimlrepo
# !pip install summarytools
# !pip install keras-tuner
```

In [3]:
```
# Import library
from ucimlrepo import fetch_ucirepo
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
from itertools import count
from keras.layers import Dense, Dropout
import kerastuner as kt
from keras.models import Sequential
from keras.layers import Input
from keras.models import Model
from summarytools import dfSummary as pc
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import tensorflow as tf
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import ReduceLROnPlateau
import warnings
warnings.filterwarnings('ignore')
```

In [4]:
```
# fetch dataset
student_performance = fetch_ucirepo(id=320)
```

In [5]:
```
# data (as pandas dataframes)
X = student_performance.data.features
y = student_performance.data.targets
```

In [6]:
```
# variable information
print(student_performance.variables)
```

	name	role	type	demographic	description	units	missing_values
0	school	Feature	Categorical	None	student's school (binary: 'GP' - Gabriel Perei...	None	no
1	sex	Feature	Binary	Sex	student's sex (binary: 'F' - female or 'M' - m...	None	no
2	age	Feature	Integer	Age	student's age (numeric: from 15 to 22)	None	no
3	address	Feature	Categorical	None	student's home address type (binary: 'U' - urb...	None	no
4	famsize	Feature	Categorical	Other	family size (binary: 'LE3' - less or equal to ...	None	no
5	Pstatus	Feature	Categorical	Other	parent's cohabitation status (binary: 'T' - li...	None	no
6	Medu	Feature	Integer	Education Level	mother's education (numeric: 0 - none, 1 - pr...	None	no
7	Fedu	Feature	Integer	Education Level	father's education (numeric: 0 - none, 1 - pr...	None	no
8	Mjob	Feature	Categorical	Occupation	mother's job (nominal: 'teacher', 'health' car...	None	no
9	Fjob	Feature	Categorical	Occupation	father's job (nominal: 'teacher', 'health' car...	None	no
10	reason	Feature	Categorical	None	reason to choose this school (nominal: close t...	None	no
11	guardian	Feature	Categorical	None	student's guardian (nominal: 'mother', 'father...	None	no
12	traveltime	Feature	Integer	None	home to school travel time (numeric: 1 - <15 m...	None	no
13	studytime	Feature	Integer	None	weekly study time (numeric: 1 - <2 hours, 2 - ...	None	no
14	failures	Feature	Integer	None	number of past class failures (numeric: n if 1...	None	no
15	schoolsup	Feature	Binary	None	extra educational support (binary: yes or no)	None	no
16	famsup	Feature	Binary	None	family educational support (binary: yes or no)	None	no
17	paid	Feature	Binary	None	extra paid classes within the course subject (...	None	no
18	activities	Feature	Binary	None	extra-curricular activities (binary: yes or no)	None	no
19	nursery	Feature	Binary	None	attended nursery school (binary: yes or no)	None	no
20	higher	Feature	Binary	None	wants to take higher education (binary: yes or...	None	no
21	internet	Feature	Binary	None	Internet access at home (binary: yes or no)	None	no
22	romantic	Feature	Binary	None	with a romantic relationship (binary: yes or no)	None	no
23	famrel	Feature	Integer	None	quality of family relationships (numeric: from...	None	no
24	freetime	Feature	Integer	None	free time after school (numeric: from 1 - very...	None	no
25	goout	Feature	Integer	None	going out with friends (numeric: from 1 - very...	None	no
26	Dalc	Feature	Integer	None	workday alcohol consumption (numeric: from 1 -...	None	no
27	Walc	Feature	Integer	None	weekend alcohol consumption (numeric: from 1 -...	None	no
28	health	Feature	Integer	None	current health status (numeric: from 1 - very ...	None	no
29	absences	Feature	Integer	None	number of school absences (numeric: from 0 to 93)	None	no
30	G1	Target	Categorical	None	first period grade (numeric: from 0 to 20)	None	no
31	G2	Target	Categorical	None	second period grade (numeric: from 0 to 20)	None	no
32	G3	Target	Integer	None	final grade (numeric: from 0 to 20, output tar...	None	no

Data Frame Summary

In [7]:
```
pc(X)
```

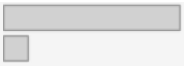
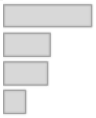
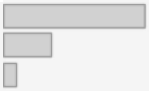
Out[7]:

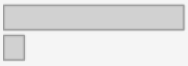
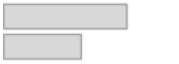
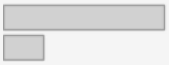
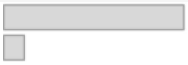
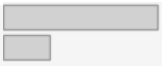
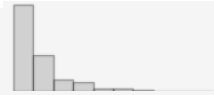
Data Frame Summary

X

Dimensions: 649 x 30

Duplicates: 0

No	Variable	Stats / Values	Freqs / (% of Valid)	Graph	Missing
1	**school** [object]	1. GP 2. MS	423 (65.2%) 226 (34.8%)		0 (0.0%)
2	**sex** [object]	1. F 2. M	383 (59.0%) 266 (41.0%)		0 (0.0%)
3	**age** [int64]	1. 17 2. 16 3. 18 4. 15 5. 19 6. 20 7. 21 8. 22	179 (27.6%) 177 (27.3%) 140 (21.6%) 112 (17.3%) 32 (4.9%) 6 (0.9%) 2 (0.3%) 1 (0.2%)		0 (0.0%)
4	**address** [object]	1. U 2. R	452 (69.6%) 197 (30.4%)		0 (0.0%)
5	**famsize** [object]	1. GT3 2. LE3	457 (70.4%) 192 (29.6%)		0 (0.0%)
6	**Pstatus** [object]	1. T 2. A	569 (87.7%) 80 (12.3%)		0 (0.0%)
7	**Medu** [int64]	1. 2 2. 4 3. 1 4. 3 5. 0	186 (28.7%) 175 (27.0%) 143 (22.0%) 139 (21.4%) 6 (0.9%)		0 (0.0%)
8	**Fedu** [int64]	1. 2 2. 1 3. 3 4. 4 5. 0	209 (32.2%) 174 (26.8%) 131 (20.2%) 128 (19.7%) 7 (1.1%)		0 (0.0%)
9	**Mjob** [object]	1. other 2. services 3. at_home 4. teacher 5. health	258 (39.8%) 136 (21.0%) 135 (20.8%) 72 (11.1%) 48 (7.4%)		0 (0.0%)
10	**Fjob** [object]	1. other 2. services 3. at_home 4. teacher 5. health	367 (56.5%) 181 (27.9%) 42 (6.5%) 36 (5.5%) 23 (3.5%)		0 (0.0%)
11	**reason** [object]	1. course 2. home 3. reputation 4. other	285 (43.9%) 149 (23.0%) 143 (22.0%) 72 (11.1%)		0 (0.0%)
12	**guardian** [object]	1. mother 2. father 3. other	455 (70.1%) 153 (23.6%) 41 (6.3%)		0 (0.0%)
13	**traveltime** [int64]	1. 1 2. 2 3. 3 4. 4	366 (56.4%) 213 (32.8%) 54 (8.3%) 16 (2.5%)		0 (0.0%)
14	**studytime** [int64]	1. 2 2. 1 3. 3 4. 4	305 (47.0%) 212 (32.7%) 97 (14.9%) 35 (5.4%)		0 (0.0%)

No	Variable	Stats / Values	Freqs / (% of Valid)	Graph	Missing
15	**failures** [int64]	1. 0 2. 1 3. 2 4. 3	549 (84.6%) 70 (10.8%) 16 (2.5%) 14 (2.2%)		0 (0.0%)
16	**schoolsup** [object]	1. no 2. yes	581 (89.5%) 68 (10.5%)		0 (0.0%)
17	**famsup** [object]	1. yes 2. no	398 (61.3%) 251 (38.7%)		0 (0.0%)
18	**paid** [object]	1. no 2. yes	610 (94.0%) 39 (6.0%)		0 (0.0%)
19	**activities** [object]	1. no 2. yes	334 (51.5%) 315 (48.5%)		0 (0.0%)
20	**nursery** [object]	1. yes 2. no	521 (80.3%) 128 (19.7%)		0 (0.0%)
21	**higher** [object]	1. yes 2. no	580 (89.4%) 69 (10.6%)		0 (0.0%)
22	**internet** [object]	1. yes 2. no	498 (76.7%) 151 (23.3%)		0 (0.0%)
23	**romantic** [object]	1. no 2. yes	410 (63.2%) 239 (36.8%)		0 (0.0%)
24	**famrel** [int64]	1. 4 2. 5 3. 3 4. 2 5. 1	317 (48.8%) 180 (27.7%) 101 (15.6%) 29 (4.5%) 22 (3.4%)		0 (0.0%)
25	**freetime** [int64]	1. 3 2. 4 3. 2 4. 5 5. 1	251 (38.7%) 178 (27.4%) 107 (16.5%) 68 (10.5%) 45 (6.9%)		0 (0.0%)
26	**goout** [int64]	1. 3 2. 2 3. 4 4. 5 5. 1	205 (31.6%) 145 (22.3%) 141 (21.7%) 110 (16.9%) 48 (7.4%)		0 (0.0%)
27	**Dalc** [int64]	1. 1 2. 2 3. 3 4. 5 5. 4	451 (69.5%) 121 (18.6%) 43 (6.6%) 17 (2.6%) 17 (2.6%)		0 (0.0%)
28	**Walc** [int64]	1. 1 2. 2 3. 3 4. 4 5. 5	247 (38.1%) 150 (23.1%) 120 (18.5%) 87 (13.4%) 45 (6.9%)		0 (0.0%)
29	**health** [int64]	1. 5 2. 3 3. 4 4. 1 5. 2	249 (38.4%) 124 (19.1%) 108 (16.6%) 90 (13.9%) 78 (12.0%)		0 (0.0%)
30	**absences** [int64]	Mean (sd) : 3.7 (4.6) min < med < max: 0.0 < 2.0 < 32.0 IQR (CV) : 6.0 (0.8)	24 distinct values		0 (0.0%)

In [8]:
```
pc(y)
```

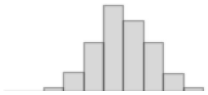
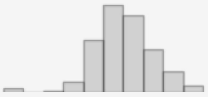
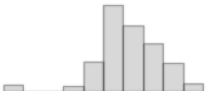
Out[8]:

Data Frame Summary

y

Dimensions: 649 x 3

Duplicates: 456

No	Variable	Stats / Values	Freqs / (% of Valid)	Graph	Missing
1	**G1** [int64]	Mean (sd) : 11.4 (2.7) min < med < max: 0.0 < 11.0 < 19.0 IQR (CV) : 3.0 (4.2)	17 distinct values		0 (0.0%)
2	**G2** [int64]	Mean (sd) : 11.6 (2.9) min < med < max: 0.0 < 11.0 < 19.0 IQR (CV) : 3.0 (4.0)	16 distinct values		0 (0.0%)
3	**G3** [int64]	Mean (sd) : 11.9 (3.2) min < med < max: 0.0 < 12.0 < 19.0 IQR (CV) : 4.0 (3.7)	17 distinct values		0 (0.0%)

In [9]:
```
y.head(15)
```

Out[9]:

	G1	G2	G3
0	0	11	11
1	9	11	11
2	12	13	12
3	14	14	14
4	11	13	13
5	12	12	13
6	13	12	13
7	10	13	13
8	15	16	17
9	12	12	13
10	14	14	14
11	10	12	13
12	12	13	12
13	12	12	13
14	14	14	15

In [10]:
```
# Convert datatype into integer
y['G1']=y['G1'].astype(int)
y['G2']=y['G2'].astype(int)
y['G3']=y['G3'].astype(int)
```

In [11]:
```
# Make A New Column Of Place Or not place
y['Placed']=y['G3'].apply(lambda x: 1 if x>=11.9 else 0)
```

In [12]:
```
y.head(5)
```

Out[12]:

	G1	G2	G3	Placed
0	0	11	11	0
1	9	11	11	0
2	12	13	12	1
3	14	14	14	1
4	11	13	13	1

In [13]:
```
# Inster y all grade data into x
X['G1'] = y['G1']
X['G2'] = y['G2']
X['G3'] = y['G3']
```

In [14]:
```
# Create the target variable by dropping unnecessary columns
y=y.drop(columns=['G1','G2','G3'])
```

In [15]:
```
y.head(5)
```

Out[15]:

	Placed
0	0
1	0
2	1
3	1
4	1

In [16]:
```
y.value_counts()
```

Out[16]:

	count
Placed	
1	348
0	301

dtype: int64

> **Placed:** 348
>
> **Not Placed:** 301

In [17]:
```
X.head(5)
```

Out[17]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel	freetime	goout	Dalc	Walc	health	absences	G1	G
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4	3	4	1	1	3	4	0	1
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	5	3	3	1	1	3	2	9	1
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	4	3	2	2	3	3	6	12	1
3	GP	F	15	U	GT3	T	4	2	health	services	...	3	2	2	1	1	5	0	14	1
4	GP	F	16	U	GT3	T	3	3	other	other	...	4	3	2	1	2	5	0	11	1

5 rows × 33 columns

Convert Text To numeric

In [18]:
```
# Convert Text To numeric
import pandas as pd
from sklearn.preprocessing import LabelEncoder, OneHotEncoder


# Label Encoding for Gender
label_encoder = LabelEncoder()
X['sex'] = label_encoder.fit_transform(X['sex'])
X['school'] = label_encoder.fit_transform(X['school'])
X['address'] = label_encoder.fit_transform(X['address'])
X['Mjob'] = label_encoder.fit_transform(X['Mjob'])
X['Fjob'] = label_encoder.fit_transform(X['Fjob'])
X['reason'] = label_encoder.fit_transform(X['reason'])

# One-Hot Encoding for Ethnicity and Age
# X = pd.get_dummies(X, columns=['Ethnicity'], prefix='Ethnicity')
X = pd.get_dummies(X, columns=['Mjob'], prefix='Mjob')
X = pd.get_dummies(X, columns=['Fjob'], prefix='Fjob')
X = pd.get_dummies(X, columns=['guardian'], prefix='guardian')
# Convert object type columns to numerical using one-hot encoding
X = pd.get_dummies(X, drop_first=True)
```

In [19]:
```
X.head(5)
```

Out[19]:

	school	sex	age	address	Medu	Fedu	reason	traveltime	studytime	failures	...	famsize_LE3	Pstatus_T	schoolsup_yes	famsup_yes	paid_ye
0	0	0	18	1	4	4	0	2	2	0	...	False	False	True	False	Fals
1	0	0	17	1	1	1	0	1	2	0	...	False	True	False	True	Fals
2	0	0	15	1	1	1	2	1	2	0	...	True	True	True	False	Fals
3	0	0	15	1	4	2	1	1	3	0	...	False	True	False	True	Fals
4	0	0	16	1	3	3	1	1	2	0	...	False	True	False	True	Fals

5 rows × 43 columns

In [20]:
```
X.describe()
```

Out[20]:

	school	sex	age	address	Medu	Fedu	reason	traveltime	studytime	failures	famrel	freeti
count	649.000000	649.000000	649.000000	649.000000	649.000000	649.000000	649.000000	649.000000	649.000000	649.000000	649.000000	649.000
mean	0.348228	0.409861	16.744222	0.696456	2.514638	2.306626	1.112481	1.568567	1.930663	0.221880	3.930663	3.180
std	0.476776	0.492187	1.218138	0.460143	1.134552	1.099931	1.192045	0.748660	0.829510	0.593235	0.955717	1.051
min	0.000000	0.000000	15.000000	0.000000	0.000000	0.000000	0.000000	1.000000	1.000000	0.000000	1.000000	1.000
25%	0.000000	0.000000	16.000000	0.000000	2.000000	1.000000	0.000000	1.000000	1.000000	0.000000	4.000000	3.000
50%	0.000000	0.000000	17.000000	1.000000	2.000000	2.000000	1.000000	1.000000	2.000000	0.000000	4.000000	3.000
75%	1.000000	1.000000	18.000000	1.000000	4.000000	3.000000	2.000000	2.000000	2.000000	0.000000	5.000000	4.000
max	1.000000	1.000000	22.000000	1.000000	4.000000	4.000000	3.000000	4.000000	4.000000	3.000000	5.000000	5.000

In [21]:
```
print(X.dtypes)
```

```
school             int64
sex                int64
age                int64
address            int64
Medu               int64
Fedu               int64
reason             int64
traveltime         int64
studytime          int64
failures           int64
famrel             int64
freetime           int64
goout              int64
Dalc               int64
Walc               int64
health             int64
absences           int64
G1                 int64
G2                 int64
G3                 int64
Mjob_0              bool
Mjob_1              bool
Mjob_2              bool
Mjob_3              bool
Mjob_4              bool
Fjob_0              bool
Fjob_1              bool
Fjob_2              bool
Fjob_3              bool
Fjob_4              bool
guardian_father     bool
guardian_mother     bool
guardian_other      bool
famsize_LE3         bool
Pstatus_T           bool
schoolsup_yes       bool
famsup_yes          bool
paid_yes            bool
activities_yes      bool
nursery_yes         bool
higher_yes          bool
internet_yes        bool
romantic_yes        bool
dtype: object
```

Scaling

In [22]:
```
from sklearn.preprocessing import MinMaxScaler

# Assuming X is your encoded data
scaler = MinMaxScaler()

# Fit and transform the data to scale it
X = scaler.fit_transform(X)
```

In [23]:
```
print(X[1])
```

```
[0.         0.         0.28571429 1.         0.25       0.25
 0.         0.         0.33333333 0.         1.         0.5
 0.5        0.         0.         0.5        0.0625     0.47368421
 0.57894737 0.57894737 1.         0.         0.         0.
 0.         0.         0.         1.         0.         0.
 1.         0.         0.         0.         1.         0.
 1.         0.         0.         0.         1.         1.
 0.        ]
```

In [24]:
```
X.shape
```

Out[24]: (649, 43)

Train-Test Split

In [25]:
```
# Split dataset
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=42)
```

Bulid Model

In [64]:
```
def build_model(hp):

  model = Sequential()

  counter = 0

  for i in range(hp.Int('num_layers', min_value=1, max_value=10)):
    if counter == 0:
      model.add(
          Dense(
              hp.Int('units' + str(i), min_value=8, max_value=512, step=16),
              activation=hp.Choice('activation' + str(i), values=['relu', 'tanh', 'sigmoid']),
              input_dim=43))
      model.add(Dropout(hp.Choice('dropout' + str(i), values=[0.1, 0.2, 0.3, 0.4, 0.5])))
    else:
      model.add(
          Dense(
              hp.Int('units' + str(i), min_value=8, max_value=512, step=16),
              activation=hp.Choice('activation' + str(i), values=['relu', 'tanh', 'sigmoid']))
          )
      model.add(Dropout(hp.Choice('dropout' + str(i), values=[0.1, 0.2, 0.3, 0.4, 0.5])))

    counter += 1
  model.add(Dense(1, activation='sigmoid'))
  model.compile(
        optimizer=hp.Choice('optimizer', values=['rmsprop', 'adam','SGD']),
        loss='binary_crossentropy',
        metrics=['accuracy']
    )
  return model
```

In [67]:
```
tunner = kt.RandomSearch(build_model,objective='val_accuracy',max_trials=10,
                         directory = 'mydir',
                         project_name = 'All_in_one3')
```

In [68]:
```
# searching the best model.
tunner.search(X_train,y_train,epochs=10,validation_data=(X_test,y_test))
```

```
Trial 10 Complete [00h 00m 21s]
val_accuracy: 0.8461538553237915

Best val_accuracy So Far: 0.8769230842590332
Total elapsed time: 00h 02m 47s
```

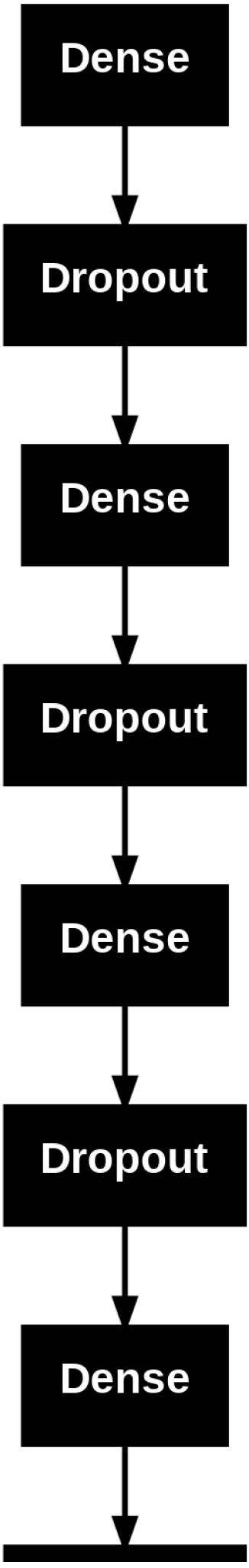
In [69]:
```
tunner.get_best_hyperparameters()[0].values
```

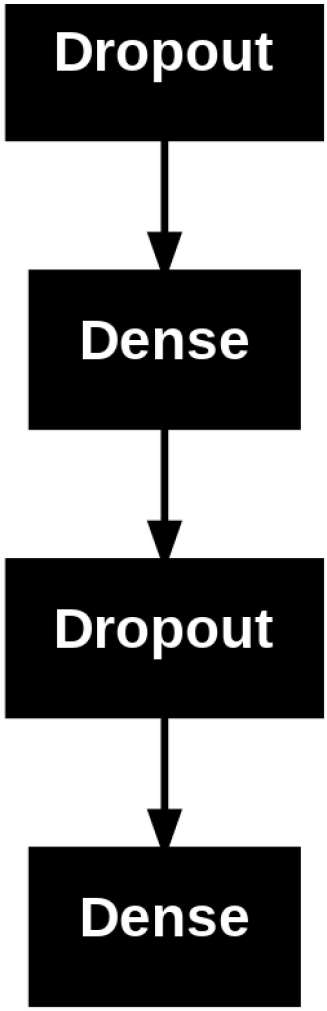
Out[69]:
```
{'num_layers': 5,
 'units0': 440,
 'activation0': 'tanh',
 'dropout0': 0.3,
 'optimizer': 'adam',
 'units1': 40,
 'activation1': 'tanh',
 'dropout1': 0.3,
 'units2': 488,
 'activation2': 'tanh',
 'dropout2': 0.2,
 'units3': 408,
 'activation3': 'tanh',
 'dropout3': 0.2,
 'units4': 72,
 'activation4': 'sigmoid',
 'dropout4': 0.3,
 'units5': 104,
 'activation5': 'relu',
 'dropout5': 0.3,
 'units6': 376,
 'activation6': 'relu',
 'dropout6': 0.2,
 'units7': 184,
 'activation7': 'sigmoid',
 'dropout7': 0.2,
 'units8': 328,
 'activation8': 'relu',
 'dropout8': 0.2,
 'units9': 104,
 'activation9': 'relu',
 'dropout9': 0.4}
```

In [78]:
```
# Take best model
model = tunner.get_best_models(num_models=1)[0]
```

In [79]:
```
from keras.utils import plot_model
plot_model(model,show_shapes=False)
```

Out[79]:





In [80]:
```python
from tensorflow.keras.callbacks import EarlyStopping

early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=10,  # No. of epochs with no improvement after which training will be stopped
    restore_best_weights=True  # Restores model weights from the epoch with the best validation loss
)

from tensorflow.keras.callbacks import ReduceLROnPlateau

reduce_lr = ReduceLROnPlateau(
    monitor='val_loss',
    factor=0.1,  # Reduce learning rate by a factor of 0.1
    patience=10,  # Number of epochs with no improvement before reducing the learning rate
    min_lr=0.0000001  # The minimum learning rate to reduce to
)
```

In [81]:
```python
# Traing the model
history = model.fit(X_train,y_train,batch_size=128,epochs=500,verbose=1,validation_data=(X_test,y_test),callbacks=[early_stopping, redu
```

```
Epoch 1/500
5/5 ━━━━━━━━━━━━━━━━━━━━ 8s 865ms/step - accuracy: 0.8617 - loss: 0.3018 - val_accuracy: 0.8538 - val_loss: 0.3096 - learning_rate: 0.00
10
Epoch 2/500
5/5 ━━━━━━━━━━━━━━━━━━━━ 4s 10ms/step - accuracy: 0.8752 - loss: 0.2838 - val_accuracy: 0.8615 - val_loss: 0.2934 - learning_rate: 0.001
0
Epoch 3/500
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.8726 - loss: 0.2787 - val_accuracy: 0.8692 - val_loss: 0.2660 - learning_rate: 0.001
0
Epoch 4/500
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.8741 - loss: 0.3019 - val_accuracy: 0.8462 - val_loss: 0.3483 - learning_rate: 0.0010
Epoch 5/500
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8649 - loss: 0.3057 - val_accuracy: 0.8692 - val_loss: 0.2604 - learning_rate: 0.001
0
Epoch 6/500
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8806 - loss: 0.2880 - val_accuracy: 0.8846 - val_loss: 0.2640 - learning_rate: 0.0010
Epoch 7/500
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.8627 - loss: 0.2834 - val_accuracy: 0.8308 - val_loss: 0.3600 - learning_rate: 0.0010
Epoch 8/500
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8521 - loss: 0.2856 - val_accuracy: 0.8615 - val_loss: 0.2881 - learning_rate: 0.0010
Epoch 9/500
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8571 - loss: 0.3258 - val_accuracy: 0.8462 - val_loss: 0.3084 - learning_rate: 0.0010
Epoch 10/500
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8890 - loss: 0.2673 - val_accuracy: 0.8692 - val_loss: 0.2780 - learning_rate: 0.001
0
Epoch 11/500
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8714 - loss: 0.2874 - val_accuracy: 0.8692 - val_loss: 0.2729 - learning_rate: 0.0010
Epoch 12/500
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.8823 - loss: 0.2516 - val_accuracy: 0.9231 - val_loss: 0.2452 - learning_rate: 0.001
0
Epoch 13/500
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - accuracy: 0.8921 - loss: 0.2510 - val_accuracy: 0.8846 - val_loss: 0.2511 - learning_rate: 0.001
0
Epoch 14/500
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.9109 - loss: 0.2305 - val_accuracy: 0.8846 - val_loss: 0.2516 - learning_rate: 0.001
0
Epoch 15/500
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.9065 - loss: 0.2234 - val_accuracy: 0.9000 - val_loss: 0.2662 - learning_rate: 0.0010
Epoch 16/500
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8250 - loss: 0.4166 - val_accuracy: 0.8077 - val_loss: 0.3509 - learning_rate: 0.0010
Epoch 17/500
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8080 - loss: 0.4088 - val_accuracy: 0.8462 - val_loss: 0.3050 - learning_rate: 0.0010
Epoch 18/500
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.8705 - loss: 0.3046 - val_accuracy: 0.8923 - val_loss: 0.2929 - learning_rate: 0.001
0
Epoch 19/500
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8853 - loss: 0.2872 - val_accuracy: 0.8000 - val_loss: 0.3501 - learning_rate: 0.0010
Epoch 20/500
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8545 - loss: 0.3161 - val_accuracy: 0.8692 - val_loss: 0.2844 - learning_rate: 0.0010
Epoch 21/500
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8983 - loss: 0.2218 - val_accuracy: 0.9000 - val_loss: 0.2369 - learning_rate: 0.001
0
Epoch 22/500
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8852 - loss: 0.2532 - val_accuracy: 0.8462 - val_loss: 0.3139 - learning_rate: 0.0010
Epoch 23/500
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.8943 - loss: 0.2526 - val_accuracy: 0.8769 - val_loss: 0.2421 - learning_rate: 0.0010
Epoch 24/500
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8891 - loss: 0.2472 - val_accuracy: 0.8923 - val_loss: 0.2285 - learning_rate: 0.001
0
Epoch 25/500
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.9075 - loss: 0.2213 - val_accuracy: 0.9077 - val_loss: 0.2035 - learning_rate: 0.001
0
Epoch 26/500
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.8952 - loss: 0.2367 - val_accuracy: 0.9154 - val_loss: 0.2113 - learning_rate: 0.001
0
Epoch 27/500
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.9002 - loss: 0.2104 - val_accuracy: 0.9154 - val_loss: 0.2304 - learning_rate: 0.0010
Epoch 28/500
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.9148 - loss: 0.2103 - val_accuracy: 0.8538 - val_loss: 0.2735 - learning_rate: 0.0010
Epoch 29/500
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8811 - loss: 0.2433 - val_accuracy: 0.9077 - val_loss: 0.2176 - learning_rate: 0.0010
Epoch 30/500
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8879 - loss: 0.2442 - val_accuracy: 0.9154 - val_loss: 0.1913 - learning_rate: 0.0010
Epoch 31/500
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.9211 - loss: 0.2160 - val_accuracy: 0.9385 - val_loss: 0.1711 - learning_rate: 0.001
0
Epoch 32/500
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8943 - loss: 0.2273 - val_accuracy: 0.9231 - val_loss: 0.1817 - learning_rate: 0.0010
Epoch 33/500
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.8818 - loss: 0.2362 - val_accuracy: 0.8615 - val_loss: 0.2370 - learning_rate: 0.0010
Epoch 34/500
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8989 - loss: 0.2151 - val_accuracy: 0.9231 - val_loss: 0.1722 - learning_rate: 0.0010
Epoch 35/500
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.8951 - loss: 0.2235 - val_accuracy: 0.8231 - val_loss: 0.3057 - learning_rate: 0.001
0
Epoch 36/500
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8727 - loss: 0.2659 - val_accuracy: 0.8923 - val_loss: 0.2241 - learning_rate: 0.0010
Epoch 37/500
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.9054 - loss: 0.2018 - val_accuracy: 0.9308 - val_loss: 0.2027 - learning_rate: 0.0010
Epoch 38/500
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.9194 - loss: 0.2107 - val_accuracy: 0.8231 - val_loss: 0.4327 - learning_rate: 0.0010
Epoch 39/500
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.8561 - loss: 0.3560 - val_accuracy: 0.9154 - val_loss: 0.2121 - learning_rate: 0.0010
Epoch 40/500
```

```
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.8868 - loss: 0.2624 - val_accuracy: 0.9231 - val_loss: 0.2043 - learning_rate: 0.0010
Epoch 41/500
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8999 - loss: 0.2254 - val_accuracy: 0.8154 - val_loss: 0.3783 - learning_rate: 0.0010
```

In [82]:
```
model.evaluate(X_test,y_test)
```

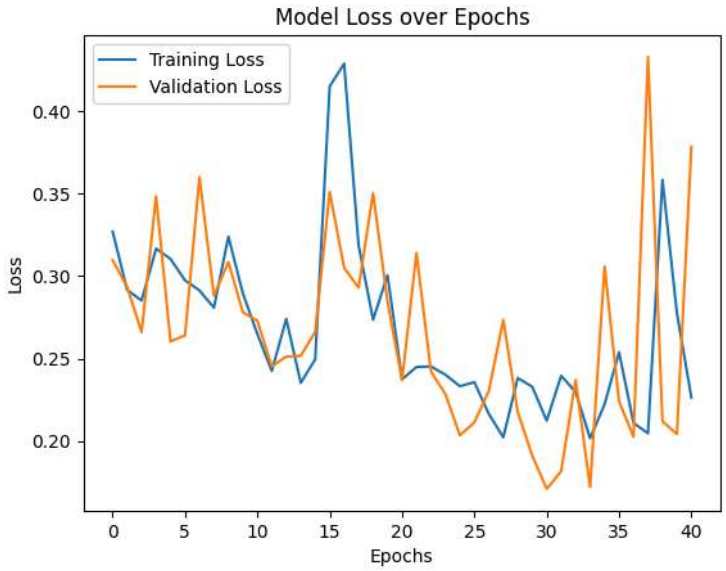
```
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9317 - loss: 0.1824
```

Out[82]:
```
[0.17106713354587555, 0.9384615421295166]
```

In [83]:
```
predictions1=model.predict(X_test)
```

```
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
```

In [84]:
```
import matplotlib.pyplot as plt

# Plotting the loss and val_loss
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')

# Adding labels and legend
plt.title('Model Loss over Epochs')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

# Show plot
plt.show()
```



In [85]:
```
#Predicted On The Test Set
y_pred= model.predict(X_test)
```

```
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
```

In [86]:
```
# Confusion Matrix
from sklearn.metrics import confusion_matrix,classification_report

# Convert predictions to binary values (adjust threshold as needed)
threshold = 0.5  # Example threshold
y_pred_binary = (y_pred > threshold).astype(int)

cm = confusion_matrix(y_test, y_pred_binary)
print(cm)
```

```
[[53  4]
 [ 4 69]]
```

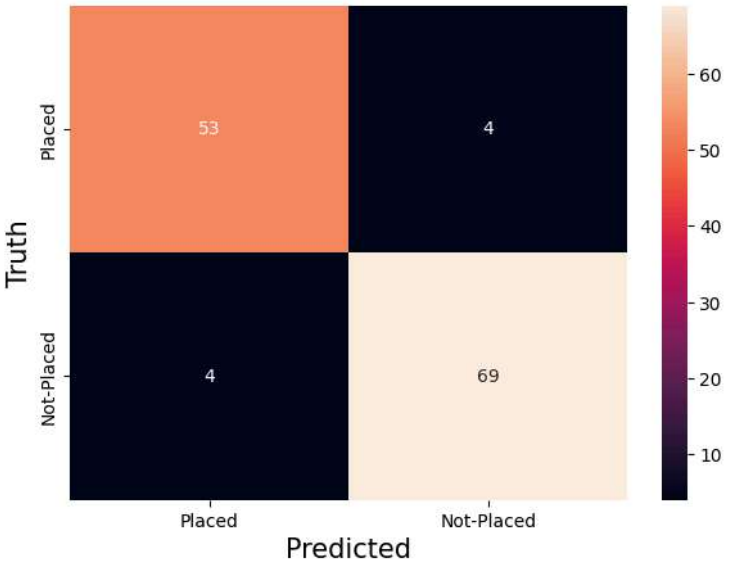
In [90]:
```
# Classification Report
print("\nClassification Report:")
classification_metrics = classification_report(y_test, y_pred_binary, target_names=['Not Place 0', 'Place 1'])
print(classification_metrics)
```

```
Classification Report:
              precision    recall  f1-score   support

 Not Place 0       0.93      0.93      0.93        57
     Place 1       0.95      0.95      0.95        73

    accuracy                           0.94       130
   macro avg       0.94      0.94      0.94       130
weighted avg       0.94      0.94      0.94       130
```

Confusion Matric

In [91]:
```
# confusion metric visual
import seaborn as sn
plt.figure(figsize=(7,5))
```

```
cm_df = pd.DataFrame(cm,
                     index = ['Placed','Not-Placed'],
                     columns = ['Placed','Not-Placed'])
sn.heatmap(cm_df,annot=True)

plt.xlabel('Predicted',color='black',size=15)
plt.ylabel('Truth',color='black',size=15)
plt.show()
```



Conclusion

The deep learning model performed well in predicting student placements, achieving an overall accuracy of 94%. The classification report indicates strong performance across both classes—students who were placed and those who were not. The model demonstrated a precision of 0.95 and recall of 0.95 for placed students, indicating that it correctly identified most of the students who secured placements. Similarly, for students who were not placed, the precision and recall were both 0.93, highlighting the model's balanced performance.

With a high F1-score for both classes (0.93 for "Not Placed" and 0.95 for "Placed"), the model effectively balances precision and recall, making it a reliable tool for predicting student placements. This result suggests that the model can be used confidently to assist in identifying placement trends and outcomes.

In [ ]: