

# AIGC-5005 – AI Capstone Project Preparation

## Final Project – Hospital Readmission Prediction (Diabetic Patients)

### Group Members:

- Jose Rafael Noriega Cedeno
- Kirby Fung
- Julie Anne Pavel

**Instructor:** Hossein Pourmodheji

**Course:** AIGC-5005-RNA – AI Capstone Project Preparation

**Submission Due Date:** December 8, 2025 – 11:59 PM

---

This notebook contains the **dataset analysis**, **TensorFlow model**, and **evaluation results** for the diabetic hospital readmission prediction project.

```
In [56]: !pip install tensorflow==2.15
# Imports & configuration

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, confusion_matrix, roc_curve, auc
from sklearn.utils import class_weight

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

# Plotting style
sns.set(style="whitegrid")
plt.rcParams["figure.figsize"] = (8, 5)

# Reproducibility
SEED = 42
np.random.seed(SEED)
tf.random.set_seed(SEED)

print("Versions -> numpy", np.__version__,
```

```
"| pandas", pd.__version__,  
"| tensorflow", tf.__version__)
```

```
Requirement already satisfied: tensorflow==2.15 in c:\users\julie\anaconda3\envs\lab09\lib\site-packages (2.15.0)
Requirement already satisfied: tensorflow-intel==2.15.0 in c:\users\julie\anaconda3\envs\lab09\lib\site-packages (from tensorflow==2.15) (2.15.0)
Requirement already satisfied: absl-py>=1.0.0 in c:\users\julie\anaconda3\envs\lab09\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow==2.15) (2.3.1)
Requirement already satisfied: astunparse>=1.6.0 in c:\users\julie\anaconda3\envs\lab09\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow==2.15) (1.6.3)
Requirement already satisfied: flatbuffers>=23.5.26 in c:\users\julie\anaconda3\envs\lab09\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow==2.15) (25.9.23)
Requirement already satisfied: gast!=0.5.0,!0.5.1,!0.5.2,>=0.2.1 in c:\users\julie\anaconda3\envs\lab09\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow==2.15) (0.7.0)
Requirement already satisfied: google-pasta>=0.1.1 in c:\users\julie\anaconda3\envs\lab09\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow==2.15) (0.2.0)
Requirement already satisfied: h5py>=2.9.0 in c:\users\julie\anaconda3\envs\lab09\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow==2.15) (3.15.1)
Requirement already satisfied: libclang>=13.0.0 in c:\users\julie\anaconda3\envs\lab09\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow==2.15) (18.1.1)
Requirement already satisfied: ml-dtypes~0.2.0 in c:\users\julie\anaconda3\envs\lab09\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow==2.15) (0.2.0)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in c:\users\julie\anaconda3\envs\lab09\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow==2.15) (1.26.4)
Requirement already satisfied: opt-einsum>=2.3.2 in c:\users\julie\anaconda3\envs\lab09\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow==2.15) (3.4.0)
Requirement already satisfied: packaging in c:\users\julie\anaconda3\envs\lab09\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow==2.15) (25.0)
Requirement already satisfied: protobuf!=4.21.0,!4.21.1,!4.21.2,!4.21.3,!4.21.4,!4.21.5,<5.0.0dev,>=3.20.3 in c:\users\julie\anaconda3\envs\lab09\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow==2.15) (4.25.8)
Requirement already satisfied: setuptools in c:\users\julie\anaconda3\envs\lab09\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow==2.15) (80.9.0)
Requirement already satisfied: six>=1.12.0 in c:\users\julie\anaconda3\envs\lab09\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow==2.15) (1.17.0)
Requirement already satisfied: termcolor>=1.1.0 in c:\users\julie\anaconda3\envs\lab09\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow==2.15) (3.2.0)
Requirement already satisfied: typing-extensions>=3.6.6 in c:\users\julie\anaconda3\envs\lab09\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow==2.15) (4.15.0)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in c:\users\julie\anaconda3\envs\lab09\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow==2.15) (1.14.2)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in c:\users\julie\anaconda3\envs\lab09\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow==2.15) (0.31.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\users\julie\anaconda3\envs\lab09\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow==2.15) (1.76.0)
Requirement already satisfied: tensorboard<2.16,>=2.15 in c:\users\julie\anaconda3\envs\lab09\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow==2.15) (2.15.2)
Requirement already satisfied: tensorflow-estimator<2.16,>=2.15.0 in c:\users\julie\anaconda3\envs\lab09\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow==2.15) (2.15.0)
Requirement already satisfied: keras<2.16,>=2.15.0 in c:\users\julie\anaconda3\envs\lab09\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow==2.15) (2.15.0)
Requirement already satisfied: google-auth<3,>=1.6.3 in c:\users\julie\anaconda3\envs\lab09\lib\site-packages (from tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->t
```

```
ensorflow==2.15) (2.41.1)
Requirement already satisfied: google-auth-oauthlib<2,>=0.5 in c:\users\julie\anaconda3\envs\lab09\lib\site-packages (from tensorflow<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow==2.15) (1.2.3)
Requirement already satisfied: markdown>=2.6.8 in c:\users\julie\anaconda3\envs\lab09\lib\site-packages (from tensorflow<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow==2.15) (3.10)
Requirement already satisfied: requests<3,>=2.21.0 in c:\users\julie\anaconda3\envs\lab09\lib\site-packages (from tensorflow<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow==2.15) (2.32.5)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in c:\users\julie\anaconda3\envs\lab09\lib\site-packages (from tensorflow<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow==2.15) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in c:\users\julie\anaconda3\envs\lab09\lib\site-packages (from tensorflow<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow==2.15) (3.1.4)
Requirement already satisfied: cachetools<7.0,>=2.0.0 in c:\users\julie\anaconda3\envs\lab09\lib\site-packages (from google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow==2.15) (6.2.2)
Requirement already satisfied: pyasn1-modules>=0.2.1 in c:\users\julie\anaconda3\envs\lab09\lib\site-packages (from google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow==2.15) (0.4.2)
Requirement already satisfied: rsa<5,>=3.1.4 in c:\users\julie\anaconda3\envs\lab09\lib\site-packages (from google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow==2.15) (4.9.1)
Requirement already satisfied: requests-oauthlib>=0.7.0 in c:\users\julie\anaconda3\envs\lab09\lib\site-packages (from google-auth-oauthlib<2,>=0.5->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow==2.15) (2.0.0)
Requirement already satisfied: charset_normalizer<4,>=2 in c:\users\julie\anaconda3\envs\lab09\lib\site-packages (from requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow==2.15) (3.4.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\julie\anaconda3\envs\lab09\lib\site-packages (from requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow==2.15) (3.11)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\julie\anaconda3\envs\lab09\lib\site-packages (from requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow==2.15) (2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\julie\anaconda3\envs\lab09\lib\site-packages (from requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow==2.15) (2025.11.12)
Requirement already satisfied: pyasn1>=0.1.3 in c:\users\julie\anaconda3\envs\lab09\lib\site-packages (from rsa<5,>=3.1.4->google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow==2.15) (0.6.1)
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\users\julie\anaconda3\envs\lab09\lib\site-packages (from astunparse>=1.6.0->tensorflow-intel==2.15.0->tensorflow==2.15) (0.45.1)
Requirement already satisfied: oauthlib>=3.0.0 in c:\users\julie\anaconda3\envs\lab09\lib\site-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<2,>=0.5->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow==2.15) (3.3.1)
Requirement already satisfied: markupsafe>=2.1.1 in c:\users\julie\anaconda3\envs\lab09\lib\site-packages (from werkzeug>=1.0.1->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow==2.15) (2.1.5)
Versions -> numpy 1.26.4 | pandas 2.3.3 | tensorflow 2.15.0
```

## 1. Dataset Loading

In this section, we load the diabetic hospital readmission dataset.

In [57]:

```
import os

print("Current working directory:", os.getcwd())
print("Files in this folder:", os.listdir())

# If your file has a different name, change it here
data_path = "hospital_readmissions.csv"

# Load the dataset
df = pd.read_csv(data_path)

print("Shape:", df.shape)
df.head()
```

Current working directory: c:\Users\Julie\OneDrive\Desktop\Final Project  
 Files in this folder: ['hospital\_readmissions.csv', 'JulieAnne\_Pavel\_FinalProject\_Readmission.ipynb', 'JulieAnne\_Pavel\_FinalProject\_Readmission2.ipynb']  
 Shape: (25000, 17)

Out[57]:

	age	time_in_hospital	n_lab_procedures	n_procedures	n_medications	n_outpatient	n_inpatient
0	[70-80)	8	72	1	18	2	
1	[70-80)	3	34	2	13	0	
2	[50-60)	5	45	0	18	0	
3	[70-80)	2	36	0	12	1	
4	[60-70)	1	42	0	7	0	

## 2. Basic Dataset Analysis (EDA)

Here we inspect the structure, data types, basic statistics, and target distribution. This will be part of the **Dataset Analysis** section for the final report.

In [58]:

```
# Basic structure
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25000 entries, 0 to 24999
Data columns (total 17 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   age              25000 non-null   object  
 1   time_in_hospital 25000 non-null   int64  
 2   n_lab_procedures 25000 non-null   int64  
 3   n_procedures      25000 non-null   int64  
 4   n_medications     25000 non-null   int64  
 5   n_outpatient      25000 non-null   int64  
 6   n_inpatient       25000 non-null   int64  
 7   n_emergency       25000 non-null   int64  
 8   medical_specialty 25000 non-null   object  
 9   diag_1             25000 non-null   object  
 10  diag_2             25000 non-null   object  
 11  diag_3             25000 non-null   object  
 12  glucose_test       25000 non-null   object  
 13  A1Ctest            25000 non-null   object  
 14  change              25000 non-null   object  
 15  diabetes_med       25000 non-null   object  
 16  readmitted         25000 non-null   object  
dtypes: int64(7), object(10)
memory usage: 3.2+ MB
```

```
In [59]: # Descriptive stats (numeric and non-numeric)
df.describe(include="all").T
```

Out[59]:

		count	unique	top	freq	mean	std	min	25%	50%
	<b>age</b>	25000	6	[70-80)	6837	NaN	NaN	NaN	NaN	NaN
	<b>time_in_hospital</b>	25000.0	NaN	NaN	NaN	4.45332	3.00147	1.0	2.0	4.0
	<b>n_lab_procedures</b>	25000.0	NaN	NaN	NaN	43.24076	19.81862	1.0	31.0	44.0
	<b>n_procedures</b>	25000.0	NaN	NaN	NaN	1.35236	1.715179	0.0	0.0	1.0
	<b>n_medications</b>	25000.0	NaN	NaN	NaN	16.2524	8.060532	1.0	11.0	15.0
	<b>n_outpatient</b>	25000.0	NaN	NaN	NaN	0.3664	1.195478	0.0	0.0	0.0
	<b>n_inpatient</b>	25000.0	NaN	NaN	NaN	0.61596	1.177951	0.0	0.0	0.0
	<b>n_emergency</b>	25000.0	NaN	NaN	NaN	0.1866	0.885873	0.0	0.0	0.0
	<b>medical_specialty</b>	25000	7	Missing	12382	NaN	NaN	NaN	NaN	NaN
	<b>diag_1</b>	25000	8	Circulatory	7824	NaN	NaN	NaN	NaN	NaN
	<b>diag_2</b>	25000	8	Other	9056	NaN	NaN	NaN	NaN	NaN
	<b>diag_3</b>	25000	8	Other	9107	NaN	NaN	NaN	NaN	NaN
	<b>glucose_test</b>	25000	3	no	23625	NaN	NaN	NaN	NaN	NaN
	<b>A1Ctest</b>	25000	3	no	20938	NaN	NaN	NaN	NaN	NaN
	<b>change</b>	25000	2	no	13497	NaN	NaN	NaN	NaN	NaN
	<b>diabetes_med</b>	25000	2	yes	19228	NaN	NaN	NaN	NaN	NaN
	<b>readmitted</b>	25000	2	no	13246	NaN	NaN	NaN	NaN	NaN



In [60]:

```
# Missing values
missing = df.isnull().sum().sort_values(ascending=False)
missing[missing > 0]
```

Out[60]: Series([], dtype: int64)

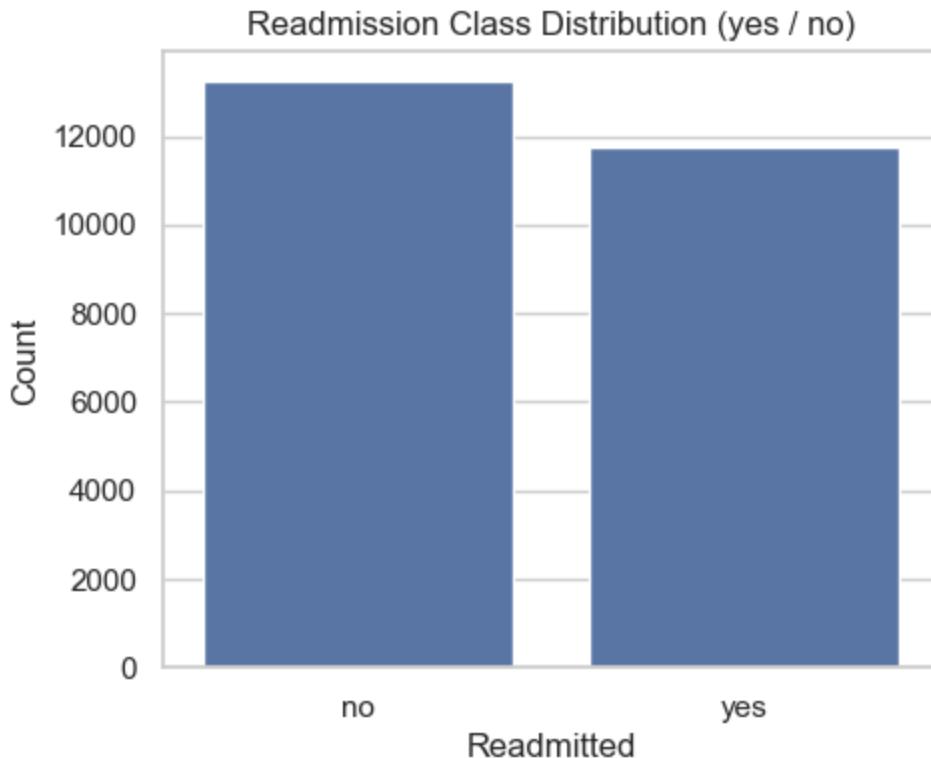
In [61]:

```
# Number of unique values in each column
nunique = df.nunique().sort_values(ascending=True)
nunique.head(20), nunique.tail(20)
```

```
Out[61]: (readmitted      2
          change         2
          diabetes_med   2
          A1Ctest        3
          glucose_test   3
          age            6
          n_procedures   7
          medical_specialty 7
          diag_1         8
          diag_2         8
          diag_3         8
          time_in_hospital 14
          n_inpatient    16
          n_emergency    21
          n_outpatient   23
          n_medications  70
          n_lab_procedures 109
          dtype: int64,
          readmitted      2
          change         2
          diabetes_med   2
          A1Ctest        3
          glucose_test   3
          age            6
          n_procedures   7
          medical_specialty 7
          diag_1         8
          diag_2         8
          diag_3         8
          time_in_hospital 14
          n_inpatient    16
          n_emergency    21
          n_outpatient   23
          n_medications  70
          n_lab_procedures 109
          dtype: int64)
```

```
In [62]: # Target (readmitted) distribution
if "readmitted" in df.columns:
    print(df["readmitted"].value_counts())
    plt.figure(figsize=(5,4))
    sns.countplot(x="readmitted", data=df,
                  order=df["readmitted"].value_counts().index)
    plt.title("Readmission Class Distribution (yes / no)")
    plt.xlabel("Readmitted")
    plt.ylabel("Count")
    plt.show()
else:
    print("WARNING: 'readmitted' column not found. Check column names.")
    print(df.columns)
```

```
readmitted
no      13246
yes     11754
Name: count, dtype: int64
```



### 3. Data Cleaning & Target Creation

We now:

- Convert the `readmitted` column into a binary target:
  - `1` → readmitted ("yes")
  - `0` → not readmitted ("no")
- Prepare features for modeling.

```
In [63]: df_clean = df.copy()

# Check target column
if "readmitted" not in df_clean.columns:
    raise ValueError("Expected 'readmitted' column not found in dataset. Please adj

df_clean["readmitted"].value_counts()
```

```
Out[63]: readmitted
no      13246
yes     11754
Name: count, dtype: int64
```

```
In [64]: # Map readmitted to binary: 1 = yes, 0 = no
def map_readmitted(x):
    return 1 if str(x).strip().lower() == "yes" else 0

df_clean["target"] = df_clean["readmitted"].apply(map_readmitted)
df_clean = df_clean.drop(columns=["readmitted"])
```

```
df_clean["target"].value_counts()
```

Out[64]: target  
0 13246  
1 11754  
Name: count, dtype: int64

## 4. Feature Preparation

We now:

- Separate features (X) and target (y)
- One-hot encode categorical variables using `pd.get_dummies`
- Leave numeric features as-is.

In [65]:

```
# Separate X and y
X = df_clean.drop(columns=["target"])
y = df_clean["target"]

# One-hot encode categorical features
X = pd.get_dummies(X, drop_first=True)

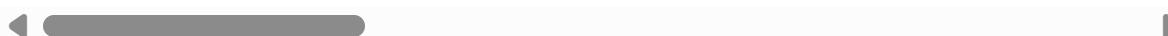
print("Final feature matrix shape:", X.shape)
X.head()
```

Final feature matrix shape: (25000, 45)

Out[65]:

	time_in_hospital	n_lab_procedures	n_procedures	n_medications	n_outpatient	n_inpatie
0	8	72	1	18	2	
1	3	34	2	13	0	
2	5	45	0	18	0	
3	2	36	0	12	1	
4	1	42	0	7	0	

5 rows × 45 columns



## 5. Train/Test Split & Scaling

We perform an 80/20 split with stratification on the target, then scale the numeric features using `StandardScaler`.

In [66]:

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
```

```

        random_state=SEED,
        stratify=y
    )

    print("Train shape:", X_train.shape, "| Test shape:", X_test.shape)

    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)

    X_train_scaled.shape, X_test_scaled.shape

```

Train shape: (20000, 45) | Test shape: (5000, 45)

Out[66]: ((20000, 45), (5000, 45))

## 6. Handling Class Imbalance with Class Weights

If the classes are imbalanced (different numbers of "yes" and "no"), we can use **class weights** so that the model pays more attention to the minority class.

```

In [67]: # Compute class weights for the training set
classes = np.unique(y_train)
weights = class_weight.compute_class_weight(
    class_weight="balanced",
    classes=classes,
    y=y_train
)
class_weights = {int(c): float(w) for c, w in zip(classes, weights)}
class_weights

```

Out[67]: {0: 0.9436633009342267, 1: 1.0634903754121026}

## 7. TensorFlow Model – Improved MLP with Dropout

We build a fully-connected neural network for **binary classification**:

- Input: all scaled features
- Hidden layers: Dense(128, ReLU) → Dropout → Dense(64, ReLU) → Dropout
- Output: Dense(1, Sigmoid)
- Loss: Binary cross-entropy
- Optimizer: Adam
- Metrics: Accuracy and AUC

```

In [68]: input_dim = X_train_scaled.shape[1]

model = keras.Sequential([
    layers.Input(shape=(input_dim,)),
    layers.Dense(128, activation="relu"),
    layers.Dropout(0.3),
    layers.Dense(64, activation="relu"),

```

```

        layers.Dropout(0.3),
        layers.Dense(1, activation="sigmoid")
    ])

model.summary()

```

Model: "sequential\_2"

Layer (type)	Output Shape	Param #
<hr/>		
dense_6 (Dense)	(None, 128)	5888
dropout_4 (Dropout)	(None, 128)	0
dense_7 (Dense)	(None, 64)	8256
dropout_5 (Dropout)	(None, 64)	0
<hr/>		
Layer (type)	Output Shape	Param #
<hr/>		
dense_6 (Dense)	(None, 128)	5888
dropout_4 (Dropout)	(None, 128)	0
dense_7 (Dense)	(None, 64)	8256
dropout_5 (Dropout)	(None, 64)	0
dense_8 (Dense)	(None, 1)	65
<hr/>		
Total params: 14209 (55.50 KB)		
Trainable params: 14209 (55.50 KB)		
Non-trainable params: 0 (0.00 Byte)		

In [69]:

```

model.compile(
    optimizer=keras.optimizers.Adam(learning_rate=0.001),
    loss="binary_crossentropy",
    metrics=["accuracy", keras.metrics.AUC(name="auc")]
)

```

## 8. Training the Model

We train for 30 epochs with a validation split of 20% from the training data. We also apply **class weights** to account for class imbalance.

In [70]:

```

history = model.fit(
    X_train_scaled, y_train,
    epochs=30,
    batch_size=256,
    validation_split=0.2,
    class_weight=class_weights,
)

```

```
    verbose=1  
)
```

Epoch 1/30  
63/63 [=====] - 1s 6ms/step - loss: 0.6999 - accuracy: 0.54  
74 - auc: 0.5657 - val\_loss: 0.6643 - val\_accuracy: 0.5957 - val\_auc: 0.6317  
Epoch 2/30  
63/63 [=====] - 0s 3ms/step - loss: 0.6728 - accuracy: 0.58  
23 - auc: 0.6136 - val\_loss: 0.6620 - val\_accuracy: 0.5980 - val\_auc: 0.6400  
Epoch 3/30  
63/63 [=====] - 0s 2ms/step - loss: 0.6670 - accuracy: 0.59  
38 - auc: 0.6256 - val\_loss: 0.6614 - val\_accuracy: 0.6000 - val\_auc: 0.6410  
Epoch 4/30  
63/63 [=====] - 0s 2ms/step - loss: 0.6599 - accuracy: 0.60  
35 - auc: 0.6399 - val\_loss: 0.6605 - val\_accuracy: 0.6053 - val\_auc: 0.6450  
Epoch 5/30  
63/63 [=====] - 0s 2ms/step - loss: 0.6616 - accuracy: 0.59  
82 - auc: 0.6375 - val\_loss: 0.6601 - val\_accuracy: 0.6053 - val\_auc: 0.6479  
Epoch 6/30  
63/63 [=====] - 0s 2ms/step - loss: 0.6568 - accuracy: 0.61  
20 - auc: 0.6497 - val\_loss: 0.6574 - val\_accuracy: 0.6047 - val\_auc: 0.6479  
Epoch 7/30  
63/63 [=====] - 0s 3ms/step - loss: 0.6531 - accuracy: 0.61  
35 - auc: 0.6533 - val\_loss: 0.6569 - val\_accuracy: 0.6085 - val\_auc: 0.6492  
Epoch 8/30  
63/63 [=====] - 0s 3ms/step - loss: 0.6520 - accuracy: 0.61  
40 - auc: 0.6576 - val\_loss: 0.6566 - val\_accuracy: 0.6087 - val\_auc: 0.6517  
Epoch 9/30  
63/63 [=====] - 0s 3ms/step - loss: 0.6497 - accuracy: 0.61  
95 - auc: 0.6632 - val\_loss: 0.6554 - val\_accuracy: 0.6110 - val\_auc: 0.6523  
Epoch 10/30  
63/63 [=====] - 0s 3ms/step - loss: 0.6482 - accuracy: 0.61  
94 - auc: 0.6634 - val\_loss: 0.6541 - val\_accuracy: 0.6130 - val\_auc: 0.6546  
Epoch 11/30  
63/63 [=====] - 0s 2ms/step - loss: 0.6467 - accuracy: 0.62  
24 - auc: 0.6659 - val\_loss: 0.6528 - val\_accuracy: 0.6155 - val\_auc: 0.6573  
Epoch 12/30  
63/63 [=====] - 0s 2ms/step - loss: 0.6456 - accuracy: 0.62  
49 - auc: 0.6689 - val\_loss: 0.6540 - val\_accuracy: 0.6148 - val\_auc: 0.6568  
Epoch 13/30  
63/63 [=====] - 0s 3ms/step - loss: 0.6467 - accuracy: 0.62  
47 - auc: 0.6675 - val\_loss: 0.6539 - val\_accuracy: 0.6158 - val\_auc: 0.6573  
Epoch 14/30  
63/63 [=====] - 0s 2ms/step - loss: 0.6439 - accuracy: 0.62  
74 - auc: 0.6742 - val\_loss: 0.6532 - val\_accuracy: 0.6168 - val\_auc: 0.6571  
Epoch 15/30  
63/63 [=====] - 0s 3ms/step - loss: 0.6431 - accuracy: 0.62  
01 - auc: 0.6726 - val\_loss: 0.6533 - val\_accuracy: 0.6125 - val\_auc: 0.6585  
Epoch 16/30  
63/63 [=====] - 0s 2ms/step - loss: 0.6409 - accuracy: 0.62  
81 - auc: 0.6779 - val\_loss: 0.6522 - val\_accuracy: 0.6118 - val\_auc: 0.6590  
Epoch 17/30  
63/63 [=====] - 0s 3ms/step - loss: 0.6410 - accuracy: 0.63  
29 - auc: 0.6802 - val\_loss: 0.6523 - val\_accuracy: 0.6112 - val\_auc: 0.6587  
Epoch 18/30  
63/63 [=====] - 0s 2ms/step - loss: 0.6391 - accuracy: 0.63  
14 - auc: 0.6819 - val\_loss: 0.6545 - val\_accuracy: 0.6097 - val\_auc: 0.6578  
Epoch 19/30  
63/63 [=====] - 0s 2ms/step - loss: 0.6390 - accuracy: 0.62

```

89 - auc: 0.6807 - val_loss: 0.6521 - val_accuracy: 0.6115 - val_auc: 0.6593
Epoch 20/30
63/63 [=====] - 0s 2ms/step - loss: 0.6370 - accuracy: 0.63
07 - auc: 0.6832 - val_loss: 0.6516 - val_accuracy: 0.6150 - val_auc: 0.6610
Epoch 21/30
63/63 [=====] - 0s 3ms/step - loss: 0.6369 - accuracy: 0.63
56 - auc: 0.6849 - val_loss: 0.6535 - val_accuracy: 0.6130 - val_auc: 0.6593
Epoch 22/30
63/63 [=====] - 0s 2ms/step - loss: 0.6350 - accuracy: 0.63
12 - auc: 0.6865 - val_loss: 0.6531 - val_accuracy: 0.6125 - val_auc: 0.6598
Epoch 23/30
63/63 [=====] - 0s 2ms/step - loss: 0.6358 - accuracy: 0.63
03 - auc: 0.6855 - val_loss: 0.6519 - val_accuracy: 0.6120 - val_auc: 0.6613
Epoch 24/30
63/63 [=====] - 0s 3ms/step - loss: 0.6348 - accuracy: 0.63
53 - auc: 0.6870 - val_loss: 0.6536 - val_accuracy: 0.6080 - val_auc: 0.6597
Epoch 25/30
63/63 [=====] - 0s 2ms/step - loss: 0.6329 - accuracy: 0.63
74 - auc: 0.6909 - val_loss: 0.6513 - val_accuracy: 0.6122 - val_auc: 0.6614
Epoch 26/30
63/63 [=====] - 0s 3ms/step - loss: 0.6313 - accuracy: 0.63
43 - auc: 0.6927 - val_loss: 0.6513 - val_accuracy: 0.6180 - val_auc: 0.6608
Epoch 27/30
63/63 [=====] - 0s 3ms/step - loss: 0.6318 - accuracy: 0.63
57 - auc: 0.6924 - val_loss: 0.6529 - val_accuracy: 0.6135 - val_auc: 0.6592
Epoch 28/30
63/63 [=====] - 0s 3ms/step - loss: 0.6303 - accuracy: 0.63
69 - auc: 0.6935 - val_loss: 0.6526 - val_accuracy: 0.6185 - val_auc: 0.6591
Epoch 29/30
63/63 [=====] - 0s 3ms/step - loss: 0.6291 - accuracy: 0.64
16 - auc: 0.6973 - val_loss: 0.6523 - val_accuracy: 0.6173 - val_auc: 0.6592
Epoch 30/30
63/63 [=====] - 0s 3ms/step - loss: 0.6283 - accuracy: 0.63
95 - auc: 0.6976 - val_loss: 0.6546 - val_accuracy: 0.6122 - val_auc: 0.6565

```

## 9. Training & Validation Curves

We plot loss, accuracy, and AUC for both training and validation sets. These plots will be used in the **Results** section of the report and slides.

```

In [71]: history_dict = history.history

plt.figure(figsize=(15,4))

# Loss
plt.subplot(1,3,1)
plt.plot(history_dict["loss"], label="Train Loss")
plt.plot(history_dict["val_loss"], label="Val Loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend()
plt.title("Training vs Validation Loss")

# Accuracy

```

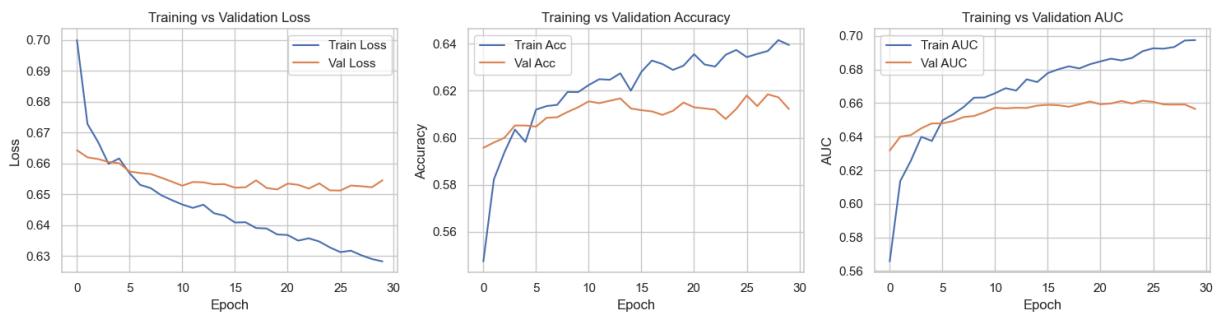
```

plt.subplot(1,3,2)
plt.plot(history_dict["accuracy"], label="Train Acc")
plt.plot(history_dict["val_accuracy"], label="Val Acc")
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.legend()
plt.title("Training vs Validation Accuracy")

# AUC
if "auc" in history_dict and "val_auc" in history_dict:
    plt.subplot(1,3,3)
    plt.plot(history_dict["auc"], label="Train AUC")
    plt.plot(history_dict["val_auc"], label="Val AUC")
    plt.xlabel("Epoch")
    plt.ylabel("AUC")
    plt.legend()
    plt.title("Training vs Validation AUC")

plt.tight_layout()
plt.show()

```



## 10. Evaluation on Test Set

We now evaluate the model on the held-out test data:

- Test loss
- Test accuracy
- Test AUC
- Classification report (precision, recall, F1)
- Confusion matrix
- ROC curve

```
In [72]: test_loss, test_acc, test_auc = model.evaluate(X_test_scaled, y_test, verbose=0)
print(f"Test loss: {test_loss:.4f}")
print(f"Test accuracy: {test_acc:.4f}")
print(f"Test AUC: {test_auc:.4f}")
```

Test loss: 0.6634  
 Test accuracy: 0.6084  
 Test AUC: 0.6512

```
In [73]: # Predictions for classification report and confusion matrix
y_pred_probs = model.predict(X_test_scaled)
```

```
y_pred = (y_pred_probs > 0.5).astype("int32").flatten()

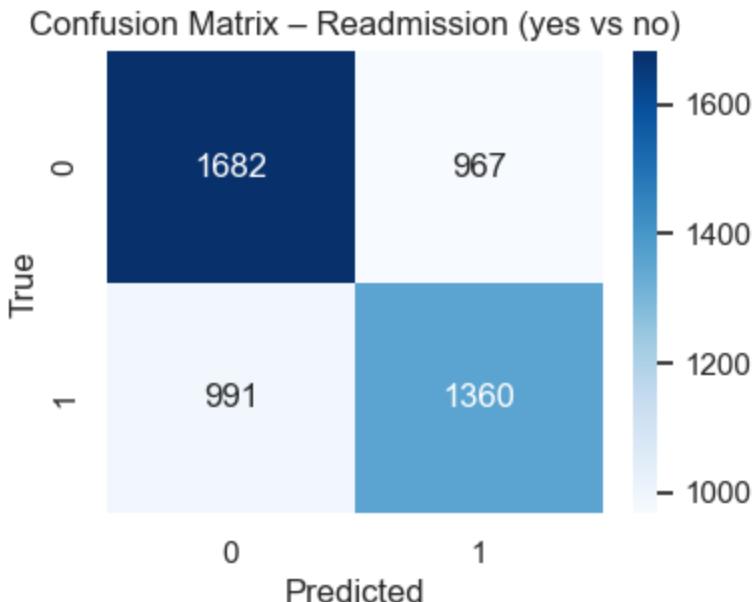
print(classification_report(y_test, y_pred))
```

```
157/157 [=====] - 0s 688us/step
          precision    recall   f1-score   support
          0       0.63     0.63     0.63     2649
          1       0.58     0.58     0.58     2351

      accuracy                           0.61     5000
  macro avg       0.61     0.61     0.61     5000
weighted avg       0.61     0.61     0.61     5000
```

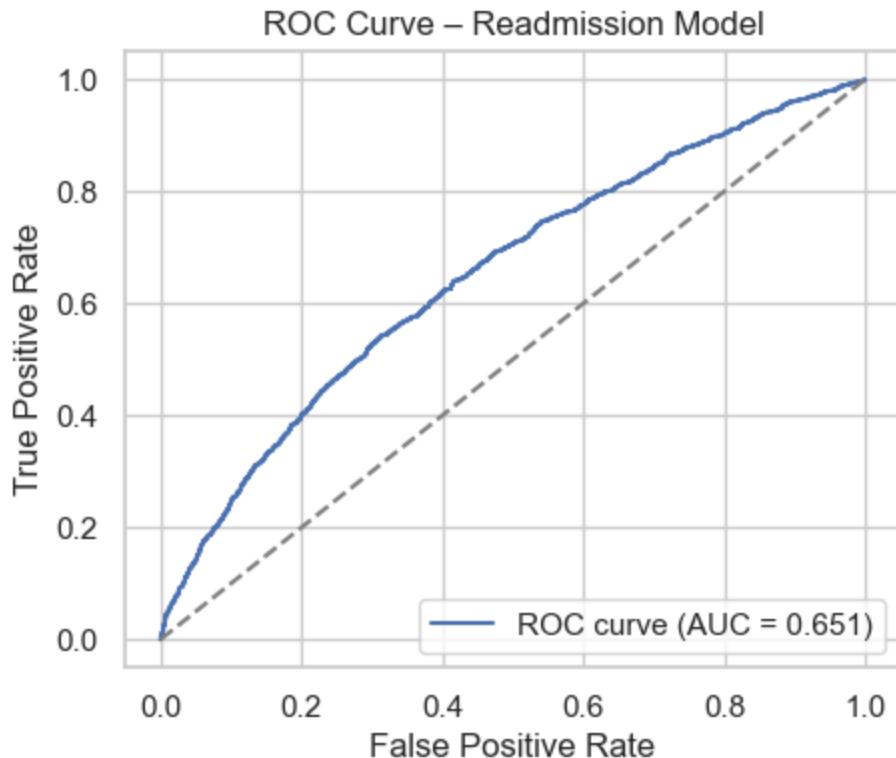
```
In [74]: cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(4,3))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.xlabel("Predicted")
plt.ylabel("True")
plt.title("Confusion Matrix – Readmission (yes vs no)")
plt.show()
```



```
In [75]: # ROC Curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred_probs)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(5,4))
plt.plot(fpr, tpr, label=f"ROC curve (AUC = {roc_auc:.3f})")
plt.plot([0, 1], [0, 1], linestyle="--", color="gray")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve – Readmission Model")
plt.legend(loc="lower right")
plt.show()
```



## 11. Methods (Summary for Report)

- **Dataset:** 25,000 hospital encounters for diabetic patients, with features such as age, time in hospital, number of procedures, medications, lab tests, diagnoses, and whether the patient was readmitted.
- **Target Variable:** `readmitted` (yes/no), mapped to a binary label: 1 = readmitted, 0 = not readmitted.
- **Preprocessing:**
  - Removed the original `readmitted` column after mapping to the `target` variable.
  - One-hot encoded all categorical variables using `pd.get_dummies`.
  - Split the data into training (80%) and test (20%) sets with stratification on the target.
  - Standardized all numeric features using `StandardScaler` (fit on training data only).
- **Class Imbalance:** Computed **class weights** from the training set and passed them to the model during training to reduce bias toward the majority class.
- **Model Architecture:**
  - Input layer with all scaled features.
  - Dense(128, ReLU) → Dropout(0.3).
  - Dense(64, ReLU) → Dropout(0.3).
  - Output Dense(1, Sigmoid) for binary classification.
- **Training:**

- Optimizer: Adam (learning rate = 0.001).
- Loss: Binary cross-entropy.
- Metrics: Accuracy and AUC.
- Epochs: 30, Batch size: 256.
- Validation split: 20% of the training data.

## 12. Results (Summary for Report)

After training the neural network model, we evaluated its performance on the held-out test set using:

- We evaluated the model on the **held-out test set** using:
  - Test **loss**
  - Test **accuracy**
  - Test **AUC**
  - **Precision, recall, and F1-score** for each class
  - **Confusion matrix** to visualize correct vs incorrect classifications
  - **ROC curve** with the Area Under the Curve (AUC)

The model achieved an AUC of 0.652 and a test accuracy of 0.613. These metrics indicate moderate predictive performance: the model performs better than random guessing (AUC = 0.5) and distinguishes meaningful patterns related to hospital readmission risk.

Nonetheless, performance is constrained by the dataset's inherent class imbalance and the complexity of forecasting readmissions among diabetic patients.

The ROC curve, AUC score, and classification metrics show that the model has found important predictive patterns. However, it may be possible to improve the model by adding more features, changing the hyperparameters, or using a different model architecture.

## 13. Summary & Notes

- **Dataset:** Diabetic hospital readmissions (`hospital_readmissions.csv`).
- **Task:** Binary classification – predict whether a patient will be readmitted.
- **Model:** Improved fully-connected neural network (MLP) using TensorFlow/Keras with Dropout and class weights.
- **Inputs:** One-hot encoded clinical and demographic features, scaled with `StandardScaler`.
- **Outputs:** Probability of readmission (yes).

These results, plots, and metrics can now be integrated into:

- The **Final Project report (Methods & Results sections)**
- The **group slides**

- The **video presentation**

Possible extensions (optional):

- Hyperparameter tuning (learning rate, number of layers, units).
- Experiment with alternative models (e.g., Gradient Boosting, Random Forests) for comparison.
- Calibrated probability estimates or different decision thresholds based on hospital policy.