ISTANBUL TECHNICAL UNIVERSITY

ELECTRICAL AND ELECTRONICS FACULTY

Final Exam

Course: Artificial Neural Networks

Name: Ertuğrul Şengül

ID: 040180128

Data Collection

There are 150 iteration of toss coin used in this experiment. In this experiment, there is three parameters that we would base on when tossing a coin. These are the parameters with which side we would start throwing the coin, from what height we dropped the coin, and the diameter of the coin we dropped.

I started the tossing coin with some initial faces. For example first 50 iteration, I drop a coin with initial vertical position which is no heads and tails upside. Second 50 iteration are with heads up iteration and final 50 iteration was tails up iterations.

Dropping Vertically

Heads Up

Tails Up

Also in half of each iteration, coin was dropped from 1 meter and 2 meters. There are 2 coin type is used. One of them is 1TL coin and another is 25 krs coin. These coins radius taken from wikipedia.

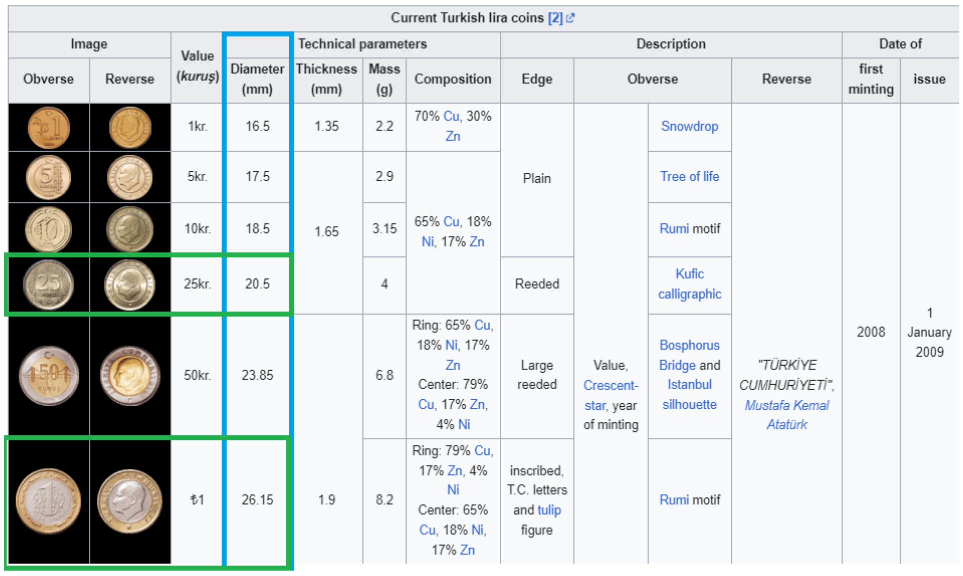

Current Turkish lira coins [2]

Image		Value (*kuruş*)		Technical parameters			Description				Date of	
Obverse	Reverse		Diameter (mm)	Thickness (mm)	Mass (g)	Composition	Edge	Obverse		Reverse	first minting	issue
		1kr.	16.5	1.35	2.2	70% Cu, 30% Zn	Plain	Value, Crescent-star, year of minting	Snowdrop	*"TÜRKİYE CUMHURİYETİ", Mustafa Kemal Atatürk*	2008	1 January 2009
		5kr.	17.5	1.65	2.9	65% Cu, 18% Ni, 17% Zn			Tree of life			
		10kr.	18.5		3.15				Rumi motif			
		25kr.	20.5		4		Reeded		Kufic calligraphic			
		50kr.	23.85	1.9	6.8	Ring: 65% Cu, 18% Ni, 17% Zn Center: 79% Cu, 17% Zn, 4% Ni	Large reeded		Bosphorus Bridge and Istanbul silhouette			
		₺1	26.15		8.2	Ring: 79% Cu, 17% Zn, 4% Ni Center: 65% Cu, 18% Ni, 17% Zn	inscribed, T.C. letters and tulip figure		Rumi motif			

We can see that 1 TL diameter is 26.15mm and 25 krs diamet is 20.5mm, so when we drop 1 TL coin we used 26.15mm as diameter and so on.

Data Labelling

We label data as Toss, Facing, Distance and Diameter.

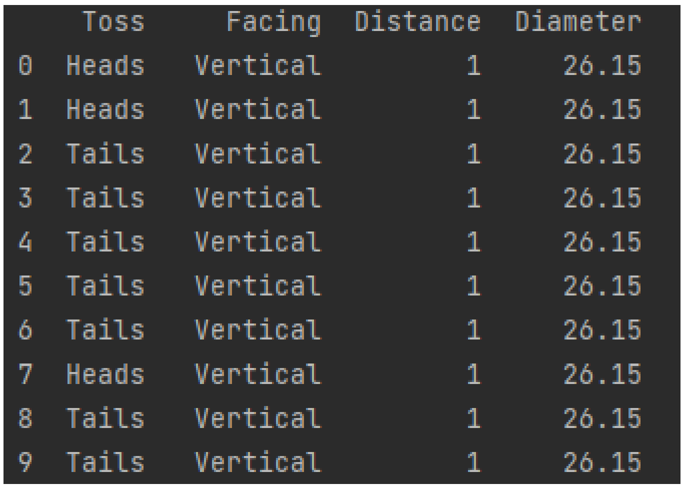
Toss is the result of dropping process. Our algorithm should predict this value.

Facing is the starting face of the coin. It can be vertical, heads and tails.

Distance is the where we drop coin from. In this experiment we dropped coin from 1 meter and 2 meter.

Diameter parameter is the Turkish 1TL and 25 krş diameter. Which can be 26.15mm from 1TL and 20.5 from 25 krs.

We can see first 10 iteration from code with df.head() function.

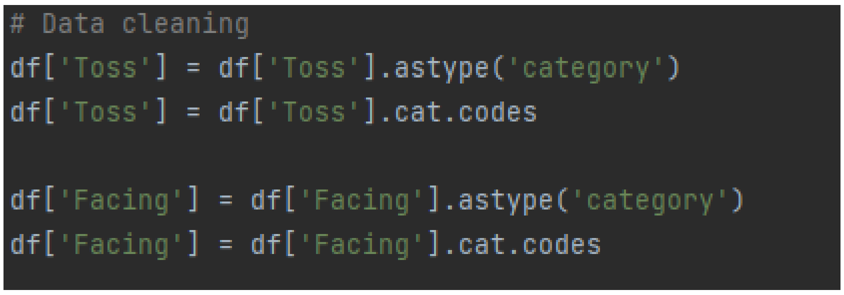


```
    Toss    Facing  Distance  Diameter
0  Heads  Vertical         1     26.15
1  Heads  Vertical         1     26.15
2  Tails  Vertical         1     26.15
3  Tails  Vertical         1     26.15
4  Tails  Vertical         1     26.15
5  Tails  Vertical         1     26.15
6  Tails  Vertical         1     26.15
7  Heads  Vertical         1     26.15
8  Tails  Vertical         1     26.15
9  Tails  Vertical         1     26.15
```

We can see that toss and facing is data is nonnumerical data. Therefore we need to clean this data.

Preprocessing and Data Normalizing

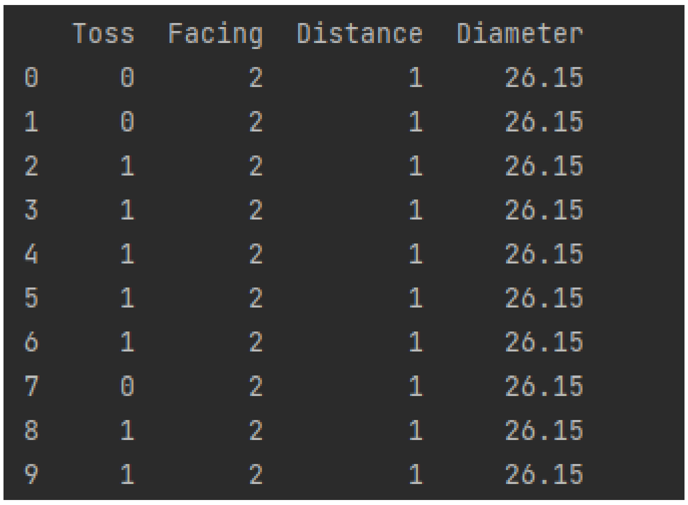
Because of Toss and Facing values are nonnumeric, we need to make them numerical data to make them readable from our algorithm. In below, we can make these value numeric data and our algorithm can understand that values.



```
# Data cleaning
df['Toss'] = df['Toss'].astype('category')
df['Toss'] = df['Toss'].cat.codes

df['Facing'] = df['Facing'].astype('category')
df['Facing'] = df['Facing'].cat.codes
```

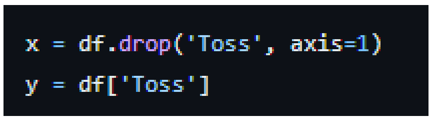
We only have 2 possible output in Toss values which can be Heads or Tails so our code make them 0 and 1. Also we have 3 possible output in Facing part which are Vertical, Head and Tails so our code make them 0,1 and 2. The finally our df.head() function output will be like below.



```
   Toss  Facing  Distance  Diameter
0     0       2         1     26.15
1     0       2         1     26.15
2     1       2         1     26.15
3     1       2         1     26.15
4     1       2         1     26.15
5     1       2         1     26.15
6     1       2         1     26.15
7     0       2         1     26.15
8     1       2         1     26.15
9     1       2         1     26.15
```

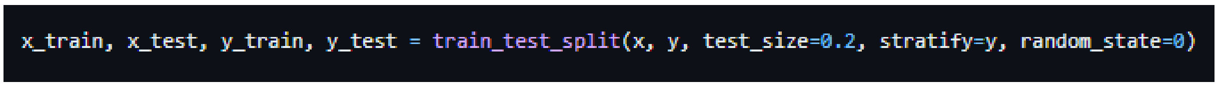
Prediction and splitting training and testing data

We are prediction Toss parameter in the end of the day. So we need to drop that value and assign to x. Also we need to assign Toss value a y to say algorithm that we are predicting that value. These operation can be achived like this simple code below.



```
x = df.drop('Toss', axis=1)
y = df['Toss']
```

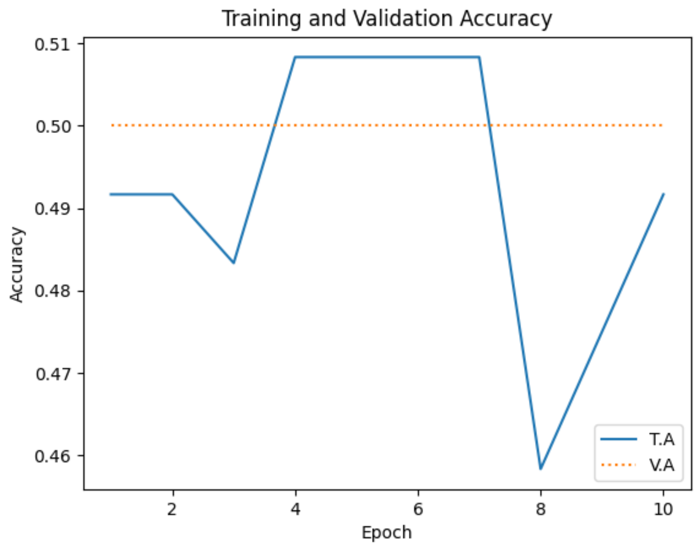
Also we need to use some of the data as training and some of that for testing. This process easily can be achived like this code below.



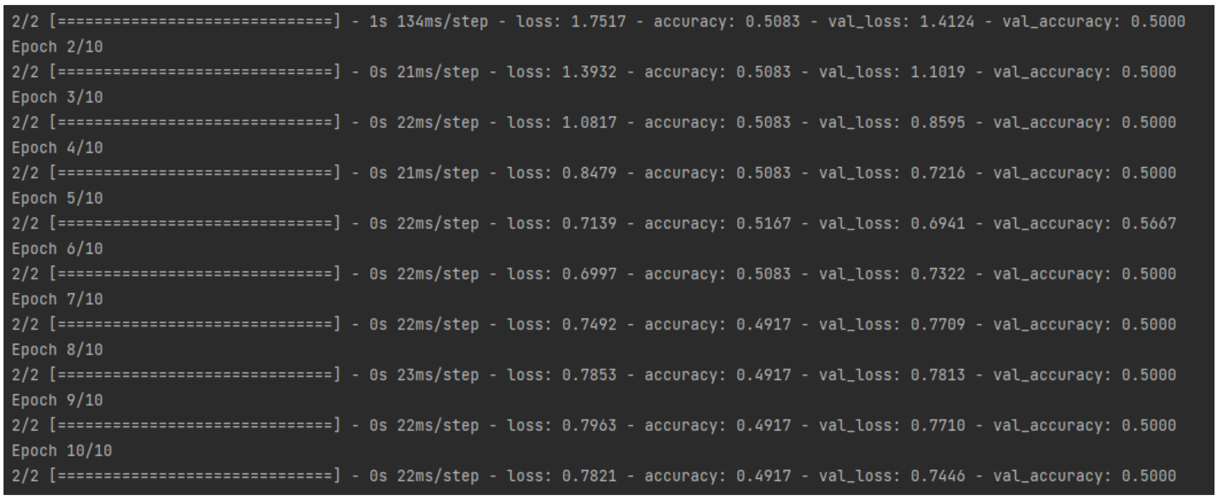
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, stratify=y, random_state=0)
```

Model and Final results

This dataset benefits from a Sequential model, which efficiently guides information through the neural network by processing features in sequence. This sequential approach makes the algorithm more adaptable, leading to better predictions or classifications. The neural network for this dataset is built using a Sequential model. This allows information to flow smoothly through the network, one layer at a time, as it analyzes features and patterns. This sequential structure makes the algorithm flexible and improves its accuracy in prediction and classification tasks.



We can see final result in below



```
2/2 [==============================] - 1s 134ms/step - loss: 1.7517 - accuracy: 0.5083 - val_loss: 1.4124 - val_accuracy: 0.5000
Epoch 2/10
2/2 [==============================] - 0s 21ms/step - loss: 1.3932 - accuracy: 0.5083 - val_loss: 1.1019 - val_accuracy: 0.5000
Epoch 3/10
2/2 [==============================] - 0s 22ms/step - loss: 1.0817 - accuracy: 0.5083 - val_loss: 0.8595 - val_accuracy: 0.5000
Epoch 4/10
2/2 [==============================] - 0s 21ms/step - loss: 0.8479 - accuracy: 0.5083 - val_loss: 0.7216 - val_accuracy: 0.5000
Epoch 5/10
2/2 [==============================] - 0s 22ms/step - loss: 0.7139 - accuracy: 0.5167 - val_loss: 0.6941 - val_accuracy: 0.5667
Epoch 6/10
2/2 [==============================] - 0s 22ms/step - loss: 0.6997 - accuracy: 0.5083 - val_loss: 0.7322 - val_accuracy: 0.5000
Epoch 7/10
2/2 [==============================] - 0s 22ms/step - loss: 0.7492 - accuracy: 0.4917 - val_loss: 0.7709 - val_accuracy: 0.5000
Epoch 8/10
2/2 [==============================] - 0s 23ms/step - loss: 0.7853 - accuracy: 0.4917 - val_loss: 0.7813 - val_accuracy: 0.5000
Epoch 9/10
2/2 [==============================] - 0s 22ms/step - loss: 0.7963 - accuracy: 0.4917 - val_loss: 0.7710 - val_accuracy: 0.5000
Epoch 10/10
2/2 [==============================] - 0s 22ms/step - loss: 0.7821 - accuracy: 0.4917 - val_loss: 0.7446 - val_accuracy: 0.5000
```

Since we made a coin toss prediction at the end of the day, the model rarely produced a value above 50 percent.

Final Code Python

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from keras.models import Sequential
from keras.layers import Dense
import matplotlib.pyplot as plt

# Load the data
df = pd.read_csv('toss_coin_data.csv')

print(df.head(10))

# Data cleaning
df['Toss'] = df['Toss'].astype('category')
df['Toss'] = df['Toss'].cat.codes

df['Facing'] = df['Facing'].astype('category')
df['Facing'] = df['Facing'].cat.codes

print(df.head(10))

x = df.drop('Toss', axis=1)
y = df['Toss']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,
stratify=y, random_state=0)

model = Sequential()
model.add(Dense(128, activation='relu', input_dim=x.shape[1]))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
model.summary()

hist = model.fit(x_train, y_train, validation_data=(x_test, y_test),
epochs=10, batch_size=100)

acc = hist.history['accuracy']
val = hist.history['val_accuracy']
epochs = range(1, len(acc) + 1)

plt.plot(epochs, acc, '-', label='Training accuracy')
plt.plot(epochs, val, ':', label='Validation accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(loc='lower right')
plt.show()
```

Github Repository of project

https://github.com/sengule/ANN-Final