```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from sklearn.preprocessing import MinMaxScaler
from matplotlib import pyplot as plt
from scipy import linalg
```

```
df = pd.read_csv("cs-training.csv", index_col = 0)
print(len(df))
df = df.dropna()
print(len(df))
X_cols = list(df.columns)
X_cols.remove("SeriousDlqin2yrs")
X = df[X_cols].to_numpy()
y = df["SeriousDlqin2yrs"].to_numpy()
```

```
150000
120269
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

Fisher LDA

```
mu = np.mean(X_train)
X_train_demeaned = (X_train - mu).T
X_test_demeaned = (X_test - mu).T
S_t = np.cov(X_train_demeaned)
S_w = np.zeros(S_t.shape)
for c in np.unique(y_train):
    S_w += np.cov(X_train_demeaned[:, y_train == c])

S_b = S_t - S_w


vals, vecs = linalg.eig(np.linalg.inv(S_w)@S_b)
vecs = vecs[:, np.argsort(vals)]
W_lda = vecs[:, -1:].real

X_train_lda = (W_lda.T@X_train_demeaned).T
X_test_lda = (W_lda.T@X_test_demeaned).T

print(np.mean(X_train_lda[y_train == 0]), np.mean(X_train_lda[y_train == 1]))
print(np.mean(X_test_lda[y_test == 0]), np.mean(X_test_lda[y_test == 1]))
```

```
85.44373421739455 83.43138114569696
85.49241455143037 83.98570707600497
```

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression(random_state=0, max_iter = 1000).fit(X_train, y_train)
probs = clf.predict_proba(X_train)
score = probs[:, 1]/(1 - probs[:, 1])
print(np.mean(score[y_train == 0]), np.mean(score[y_train == 1]))

probs = clf.predict_proba(X_test)
score = probs[:, 1]/(1 - probs[:, 1])
print(np.mean(score[y_test == 0]), np.mean(score[y_test == 1]))
```

```
0.08943248579736342 0.31403573504498794
0.0811116011885045 0.26534977342014043
```

Random Forest

```
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(max_depth=5, random_state=0)
clf.fit(X_train, y_train)
probs = clf.predict_proba(X_train)
score = probs[:, 1]/(1 - probs[:, 1])
print(np.mean(score[y_train == 0]), np.mean(score[y_train == 1]))

probs = clf.predict_proba(X_test)
score = probs[:, 1]/(1 - probs[:, 1])
print(np.mean(score[y_test == 0]), np.mean(score[y_test == 1]))
```

```
0.07023938518560487 0.39612069415283807
0.07135186453471327 0.3791356430665598
```

Dense NN

```
model_in = keras.Input(shape = (10,))
x = layers.Dense(5, activation= "relu")(model_in)
x = layers.Dense(2, activation= "relu")(x)
out = layers.Dense(1, activation= "sigmoid")(x)

model = keras.Model(model_in, out)
model.compile(optimizer='adam', loss='binary_crossentropy')
model.fit(X_train, y_train, epochs=50,
```

```
                batch_size=8,
                shuffle=True)
```

```
Epoch 1/50
10073/10073 [==============================] - 7s 636us/step - loss: 0.3964
Epoch 2/50
10073/10073 [==============================] - 6s 641us/step - loss: 0.2515
Epoch 3/50
10073/10073 [==============================] - 7s 676us/step - loss: 0.2556
Epoch 4/50
10073/10073 [==============================] - 7s 686us/step - loss: 0.2515
Epoch 5/50
10073/10073 [==============================] - 7s 690us/step - loss: 0.2515
Epoch 6/50
10073/10073 [==============================] - 7s 660us/step - loss: 0.2522
Epoch 7/50
10073/10073 [==============================] - 7s 674us/step - loss: 0.2515
Epoch 8/50
 7917/10073 [======================>.......] - ETA: 1s - loss: 0.2544
```

```
probs = model.predict(X_train)
score = probs/(1 - probs)
print(np.mean(score[y_train == 0]), np.mean(score[y_train == 1]))
```

```
0.074047506 0.074047506
```