

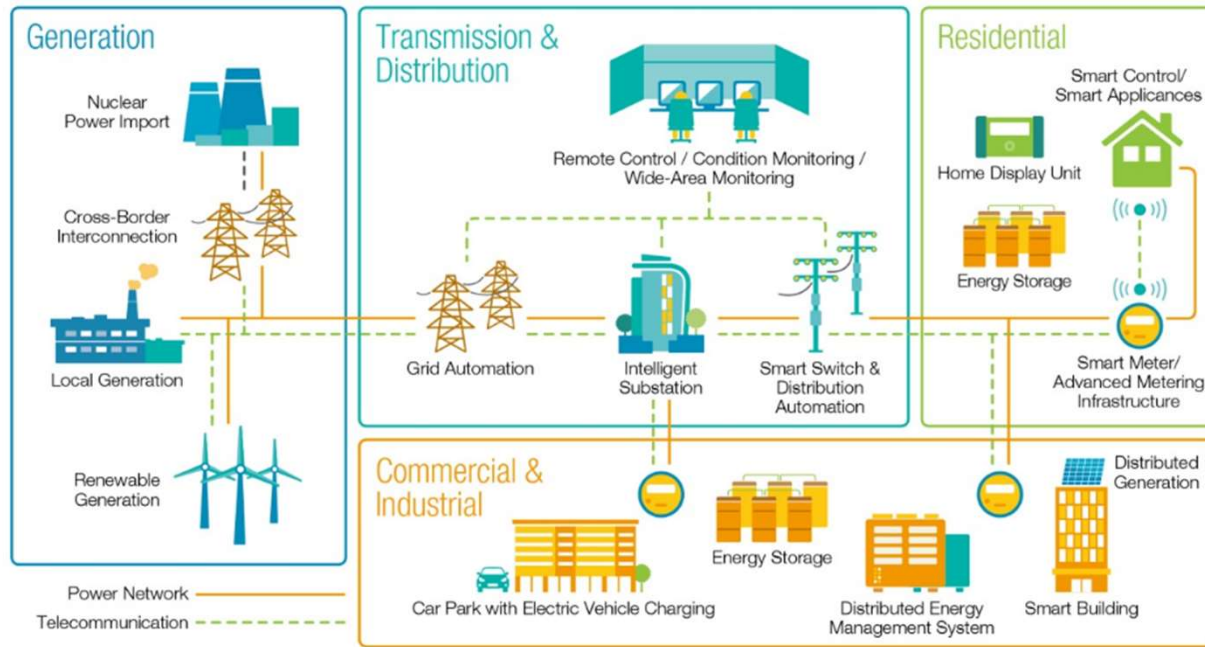
Capstone Project 3

Application of Deep Learning for the Stability Prediction of the Smart Grids

Wei Ruan

Springboard Data Science Student 2022-2023

Problem statement

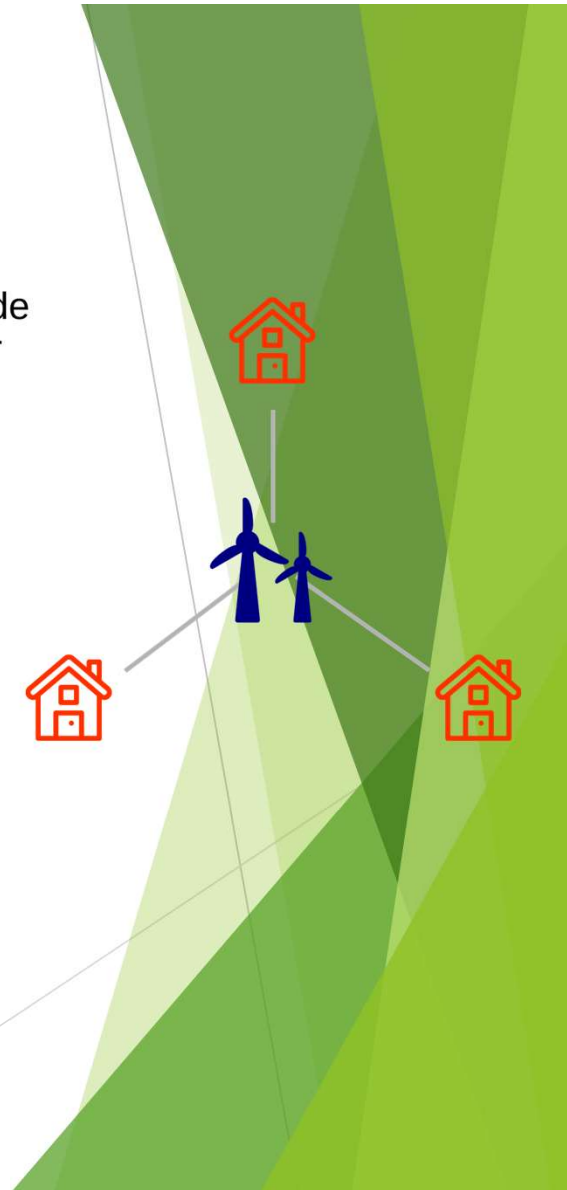
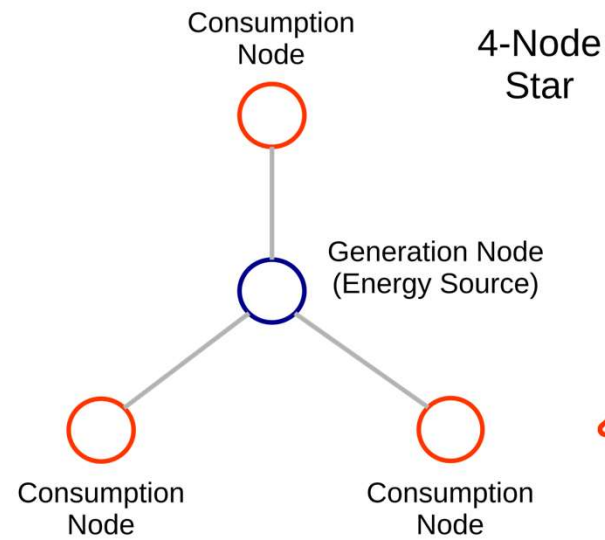


Decentral Smart Grid Control System

- Goal of this project - develop a deep learning model for a four-node-star electrical grid with a centralized production with the precision more than 90% to predict the stability of this power grid.

Dataset

- Originally provided by KIT (Karlsruher Institut für Technologie, Germany)
- 10,000 observations for 14 variables
- 12 primary features and 2 dependent variables
- About 2 months with a ten-minute interval
- Augmented data with 60,000 observations



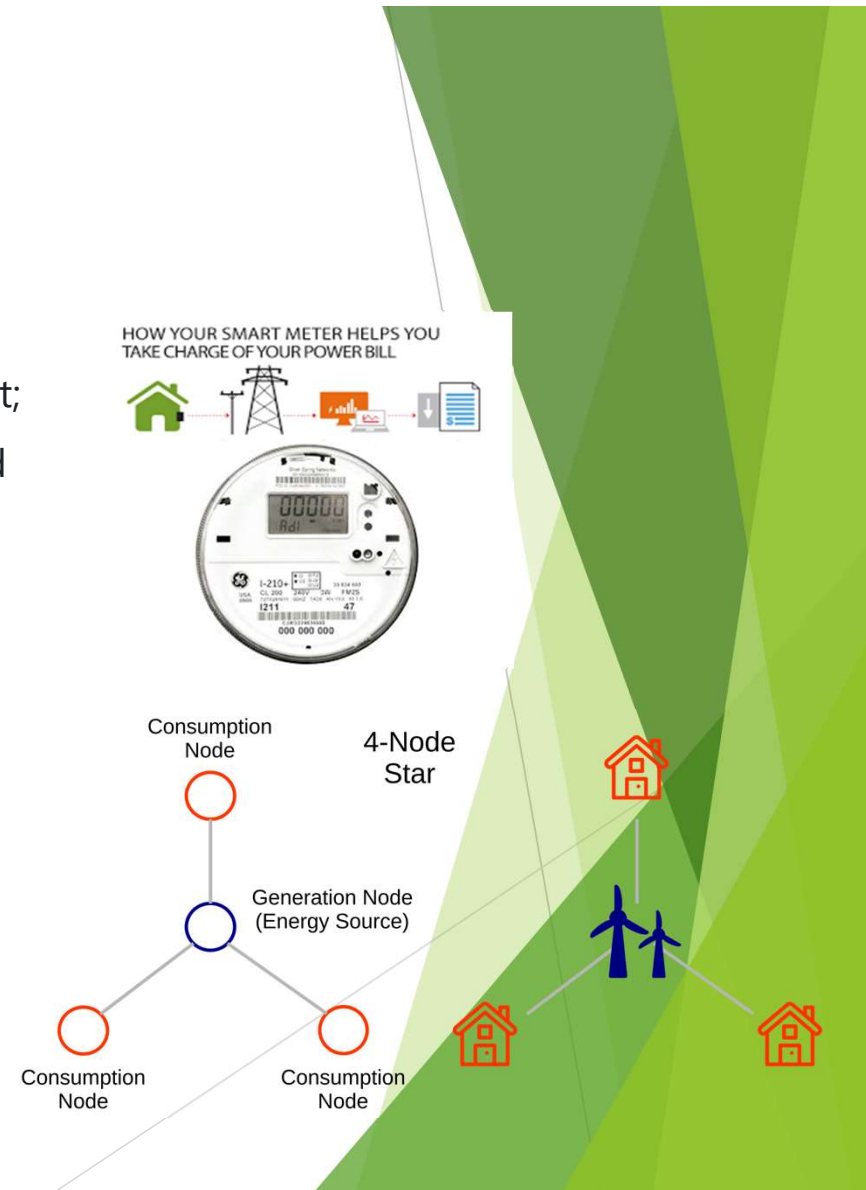
Data Structure

► Predictive features:

1. ' τ_1 ' to ' τ_4 ': the reaction time of each network participant;
2. ' p_1 ' to ' p_4 ': nominal power produced (positive) or consumed (negative) by each network participant;
3. ' g_1 ' to ' g_4 ': price elasticity coefficient for each network participant;

► Dependent variables:

1. ' stab ': the maximum real part of the characteristic differential equation root (if positive, the system is linearly unstable; if negative, linearly stable);
2. ' stabf ': a categorical (binary) label ('stable' or 'unstable').



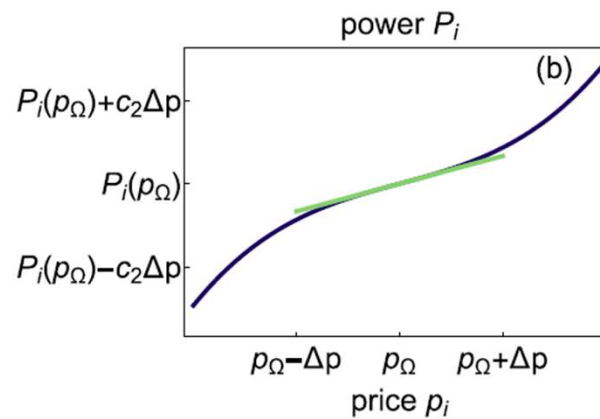
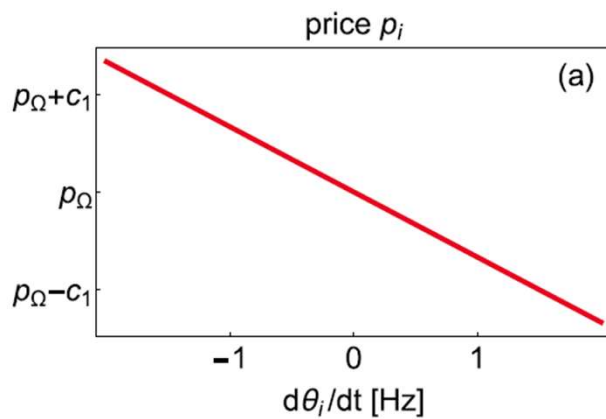
Data Wrangling

- Data relationships - tau, p, P

$$p_i \left(\frac{d\theta_i}{dt} \right) = p_\omega - c_1 \frac{d\theta_i}{dt} \quad i = 1, \dots, N. \quad (1)$$

$$\hat{P}_i(t) \approx P_i + c_2(p_i - P_\Omega) \quad i = 1, \dots, N. \quad (2)$$

$$\hat{P}_i(t) = P_i - \gamma_i \frac{d\theta_i}{dt}(t) \quad i = 1, \dots, N. \quad (3)$$



Variable Distribution



- ▶ Tau - even distribution between (0.5, 10)
- ▶ Normal power, p - p_2, p_3, p_4 even dist. $P_1 = p_2 + p_3 + p_4$, normal dist.
- ▶ Elasticity coefficient, g - even dist. Between (0, 1), $g = c_1 * c_2$
- ▶ *stab*: If $stab < 0$, the grid is stable and $stabf = "stable"$. If $stab \geq 0$, the grid is unstable and $stabf = "unstable"$

Exploratory Data Analysis (EDA)

Dependent variables

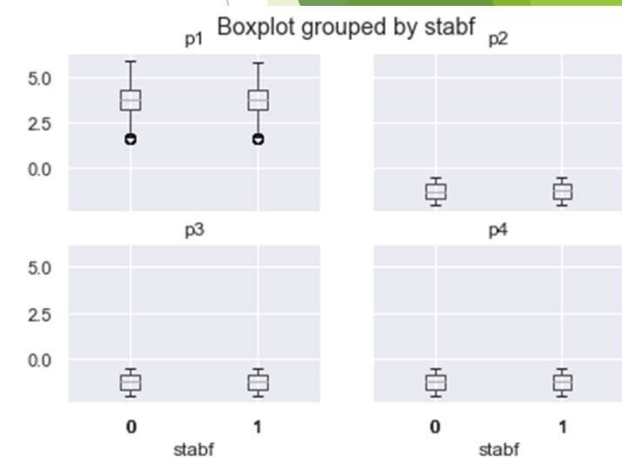
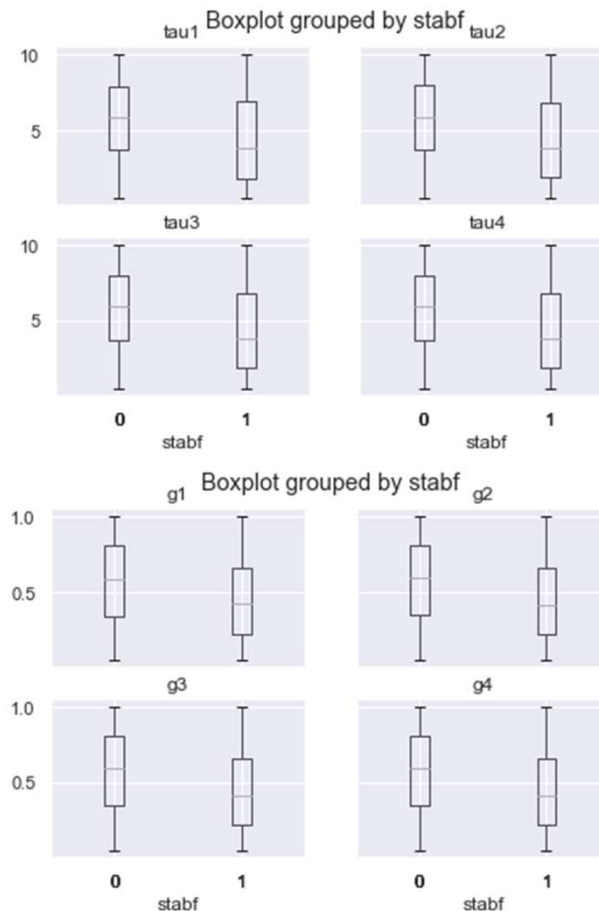
- ▶ Augmented data (60,000 observations)
- ▶ Dependent variables: stab, stabf
 1. $\text{Stab} > 0$ means $\text{stabf} = \text{unstable} (=0)$
 2. $\text{Stab} < 0$ means $\text{stabf} = \text{stable} (=1)$
- ▶ Stable ($\text{stabf}=1$) ratio = 35%



Exploratory Data Analysis (EDA)

Ranges of features

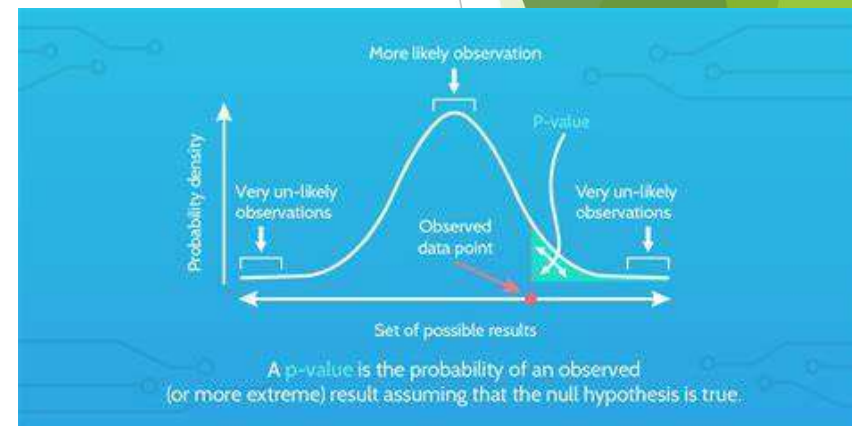
- Box plots showing the ranges of features for unstable/stable cases.
- The dist. Of tau and g are significantly different for unstable and stable cases.
- The dist. Of p is not significantly different for unstable and stable cases.



Exploratory Data Analysis (EDA)

Hypothesis tests

- ▶ H_{null} : the observed difference in the mean of $\tau_{i,i}$ ($i=1,2,3,4$) between the unstable cases and the stable cases is due to chance. That is, $\tau_{i,u} = \tau_{i,s}$
- ▶ H_{null} : $g_{i,u} = g_{i,s}$
- ▶ H_{null} : $p_{i,u} = p_{i,s}$
- ▶ H_{alter} : $\tau_{i,u} \neq \tau_{i,s}$; $g_{i,u} \neq g_{i,s}$; $p_{i,u} \neq p_{i,s}$
- ▶ P-value: the probability of obtaining results at least as extreme as the observed results assuming that the null hypothesis is correct.
- ▶ Calculate p-value for each test: p-value = 0 for test 1 and 2. p-value = 0.008 < 5% significance level for test 3.
- ▶ The null hypothesis are rejected, and the alternative ones are accepted.



Pre-processing

Correlation

- Heat map to show the correlations of features with stabf
- 1. Stab is 0.83 as the definition
- 2. τ_i ($i=1,2,3,4$) are -0.24
- 3. g_i ($i=1,2,3,4$) are -0.22
- 4. p_i ($i=1,2,3,4$) are -0.006.
- This verified the hypothesis tests



Pre-processing

Segregating train and test sets

- ▶ Ratio of train and test = 9:1
- ▶ Augmented data
 1. Train data: 54,000
 2. Test data: 6,000
- ▶ Original data
 1. Train data: 9,000
 2. Test data: 1,000

Feature assessing and scaling

- ▶ scaling is fitted and transformed to the training sets and transformed testing sets
- ▶ Augmented data

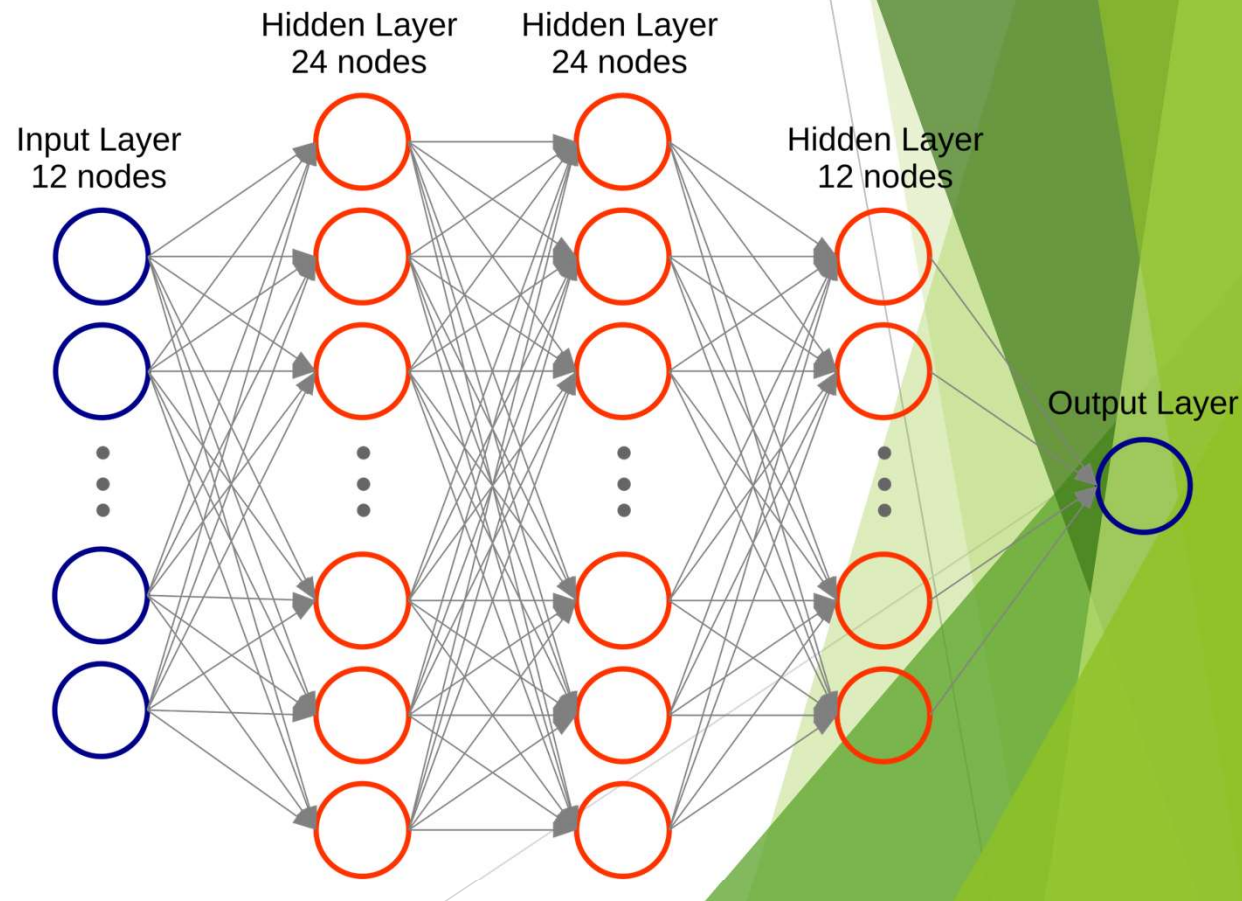
```
scaler = StandardScaler()  
X_training = scaler.fit_transform(X_training)  
X_testing = scaler.transform(X_testing)
```

- ▶ Original data

```
scaler1 = StandardScaler()  
X1_training = scaler1.fit_transform(X1_training)  
X1_testing = scaler1.transform(X1_testing)
```

Deep Learning - ANN

- ▶ One input layer (12 input nodes);
- ▶ Three hidden layers (24, 24 and 12 nodes, respectively);
- ▶ One single-node output layer.
- ▶ Choice of 'relu' as the activation function for hidden layers
- ▶ Choice of 'sigmoid' as activation for the output layers
- ▶ Choice of 'adam' as optimizer and 'binary_crossentropy' as the loss function



TensorFlow vs. keras

- ▶ TensorFlow is the most famous library for deep learning models.
- ▶ It has a very large and awesome community.
- ▶ However, TensorFlow is not that easy to use.
- ▶ Keras is a high-level API built on TensorFlow.
- ▶ It is more user-friendly and easier to use as compared to **TF**.

ANN Python Codes

- ▶ Sequential from keras lib was introduced;
- ▶ Input and 1st hidden layer added
- ▶ 2nd hidden layer added
- ▶ 3rd hidden layer added
- ▶ Single node output layer added
- ▶ Compilation with 'adam' and 'binary_crossentropy'

```
from scipy import stats
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.model_selection import KFold

from keras.models import Sequential
from keras.layers import Dense

# ANN initialization
classifier = Sequential()
# Input layer and first hidden layer
classifier.add(Dense(units = 24, kernel_initializer = 'uniform', activation = 'relu',
input_dim = 12))
# Second hidden layer
classifier.add(Dense(units = 24, kernel_initializer = 'uniform', activation = 'relu'))
# Third hidden layer
classifier.add(Dense(units = 12, kernel_initializer = 'uniform', activation = 'relu'))
# Single-node output layer
classifier.add(Dense(units = 1, kernel_initializer = 'uniform', activation = 'sigmoid'))
# ANN compilation
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
```


ANN Model Fitting

- ▶ a cross-validation based fitting is proposed.
- ▶ Kfold=10 is the cross-validation engine selected and epoch=50
- ▶ The percentage of unstable cases is 65%, so Classification Report (f1-score) was compared with the accuracy.

```
for train_index, val_index in KFold(10, shuffle=True, random_state=10).split(X_training):  
    x_train, x_val = X_training[train_index], X_training[val_index]  
    y_train, y_val = y_training[train_index], y_training[val_index]  
    classifier.fit(x_train, y_train, epochs=50, verbose=0)  
    classifier_loss, classifier_accuracy = classifier.evaluate(x_val, y_val)  
    print(f'Round {cross_val_round} - Loss: {classifier_loss:.4f} | Accuracy:  
{classifier_accuracy * 100:.2f} %')  
    cross_val_round += 1
```

```
y_pred = classifier.predict(X_testing)  
y_pred[y_pred <= 0.5] = 0  
y_pred[y_pred > 0.5] = 1  
print(classification_report(y_testing, y_pred))  
cm = pd.DataFrame(data=confusion_matrix(y_testing, y_pred, labels=[0, 1]),  
                  index=["Actual Unstable", "Actual Stable"],  
                  columns=["Predicted Unstable", "Predicted Stable"])  
  
cm  
print(f'Accuracy per the confusion matrix: {((cm.iloc[0, 0] + cm.iloc[1, 1]) /  
len(y_testing) * 100):.2f}%')
```

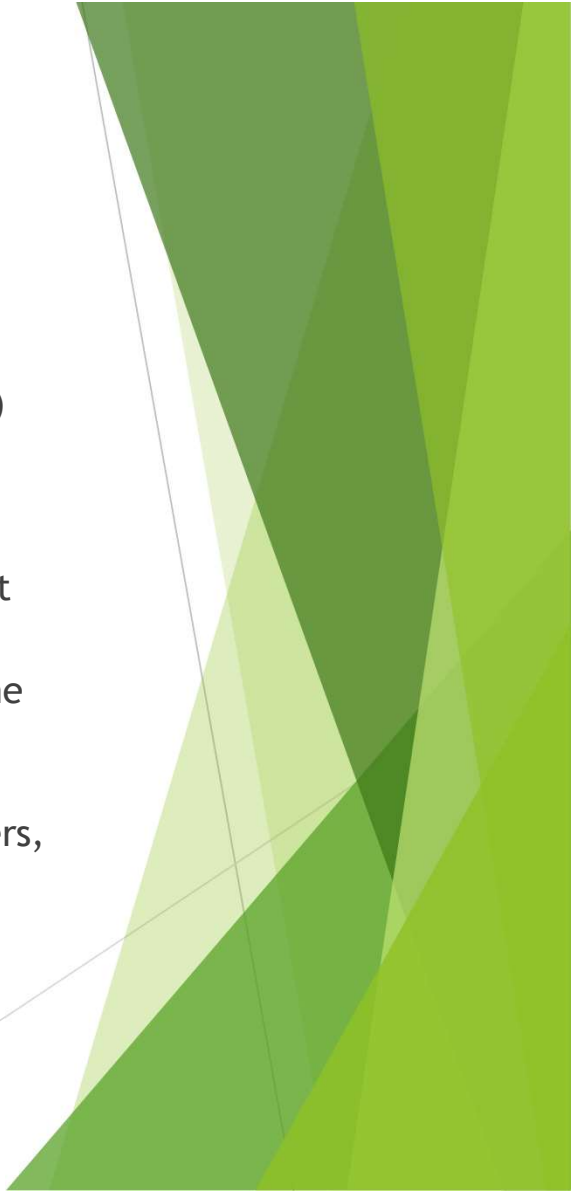
Tuning Parameters of ANN

- In addition to the base case, 11 cases for different ANN architecture
 1. 3 hidden layers vs. 2 hidden layers
 2. epoches = 10, 20, 50
 3. datasets = augmented sets vs original sets
- F1-score is true performance on both balanced and imbalanced datasets.
- F1-score considers both the precision and recall ability of the model.

Original Dataset (10,000 observations)								
Case #	Architecture	Folds	Epoches	Confusion Matrix		Accuracy (%)	F1 Score	Compute time (s)
4	24-12-1	10	20	595	14	97.2	0.98	31
				14	377			
5	24-12-1	10	20	597	12	97.4	0.98	62
				14	377			
6	24-12-1	10	50	591	18	96.9	0.97	155
				13	378			
1	24-24-12-1	10	10	601	8	98	0.99	32
				12	379			
2	24-24-12-1	10	20	598	11	97.7	0.99	62
				12	379			
3	24-24-12-1	10	50	599	10	97.7	0.98	154
				11	378			
Augmented Dataset (60,000 observations)								
Case #	Architecture	Folds	Epoches	Confusion Matrix		Accuracy (%)	F1 Score	Compute time (s)
4	24-12-1	10	20	3728	86	96.6	0.97	181
				118	2068			
5	24-12-1	10	20	3742	72	97.07	0.98	362
				104	2082			
6	24-12-1	10	50	3771	43	97.68	0.98	893
				96	2090			
1	24-24-12-1	10	10	3755	59	98.13	0.99	176
				53	2133			
2	24-24-12-1	10	20	3758	56	98.35	0.99	390
				43	2143			
3	24-24-12-1	10	50	3761	53	98.3	0.98	943
				49	2137			

A decorative graphic on the left side of the slide. It features a network of black dots connected by thin grey lines, forming a complex, web-like structure. The dots are arranged in a way that suggests a hierarchical or interconnected system, possibly representing a power grid or a neural network. The overall shape is somewhat irregular, with a denser cluster of nodes on the left and a more sparse, branching structure on the right.

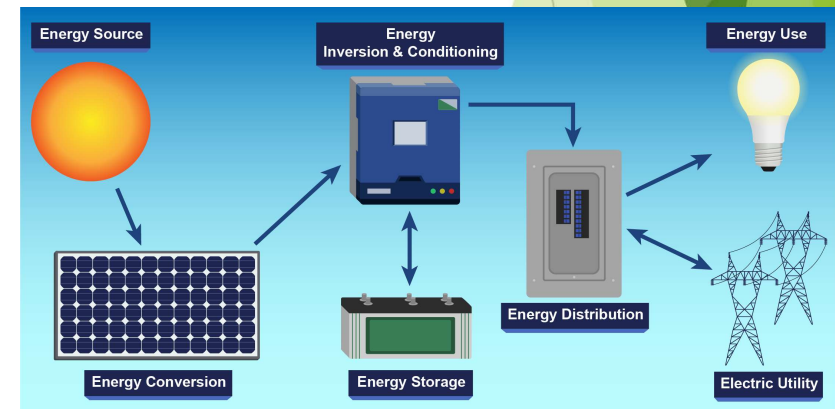
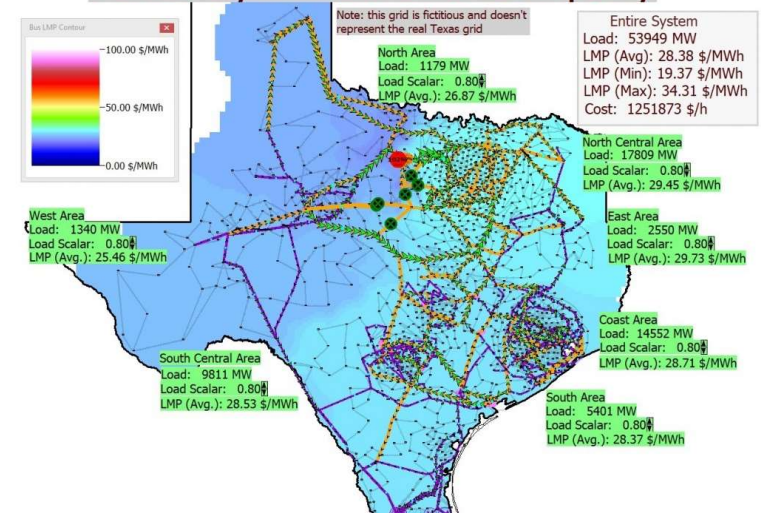
Conclusions

- ▶ All cases had the F1 Score within the range of (0.97, 0.99) and the accuracy between 96.6% - 98.3% for this classification problem.
 - ▶ The hidden layers of the architecture network had slight impacts on the F1 score (accuracy) and computing time. It seems that three layers helped the F1 score slightly.
 - ▶ The augmented dataset also increases the F1 score and the accuracy slightly. But the augmented data consume was proportional to the size of dataset.
 - ▶ For this basic power grid, the model with two hidden layers, 10 folds and 10 epoches on the augmented dataset is preferable to predict the power grid stability.
- 
- A decorative graphic on the right side of the slide. It consists of several overlapping, semi-transparent green triangles and polygons of various shades, ranging from a light lime green to a darker forest green. The shapes are arranged in a way that creates a sense of depth and movement, with some shapes appearing to be in front of others. The overall effect is a modern, abstract background element.

Future Work

- For this basic 4-node-star power grid, the ANN model has very good F1 score (accuracy) to predict the stability. To apply for this ANN model into the real operation of power grids, we still have more challenging tasks to solve.
1. More nodes of power plants and end users
 2. Solar energy implementation in the commercial and residential users
 3. More battery-based electricity storage implementation

Texas Synthetic Grid Company



BACK UP SLIDE - PERF. METRICS

		Predicted condition	
		Positive (PP)	Negative (PN)
Actual condition	Total population = P + N		
	Positive (P)	True positive (TP), hit	False negative (FN), type II error, miss, underestimation
	Negative (N)	False positive (FP), type I error, false alarm, overestimation	True negative (TN), correct rejection
	Prevalence $= \frac{P}{P + N}$	Positive predictive value (PPV), precision $= \frac{TP}{PP} = 1 - FDR$	False omission rate (FOR) $= \frac{FN}{PN} = 1 - NPV$
	Accuracy (ACC) $= \frac{TP + TN}{P + N}$	False discovery rate (FDR) $= \frac{FP}{PP} = 1 - PPV$	Negative predictive value (NPV) $= \frac{TN}{PN}$ $= 1 - FOR$
	Balanced accuracy (BA) $= \frac{TPR + TNR}{2}$	F_1 score $= \frac{2PPV \times TPR}{PPV + TPR} = \frac{2TP}{2TP + FP + FN}$	Fowlkes–Mallows index (FM) $= \sqrt{PPV \times TPR}$