AI VIETNAM
All-in-One Course
(TA Session)

Medical Data Science on Time-series Data

TA - Thanh Huy

STA – Hoang Thien

Outline

- Introduction to Medical Time Series Data
- Death Forecasting in Covid-19
- Diabete Classification
- Q&A

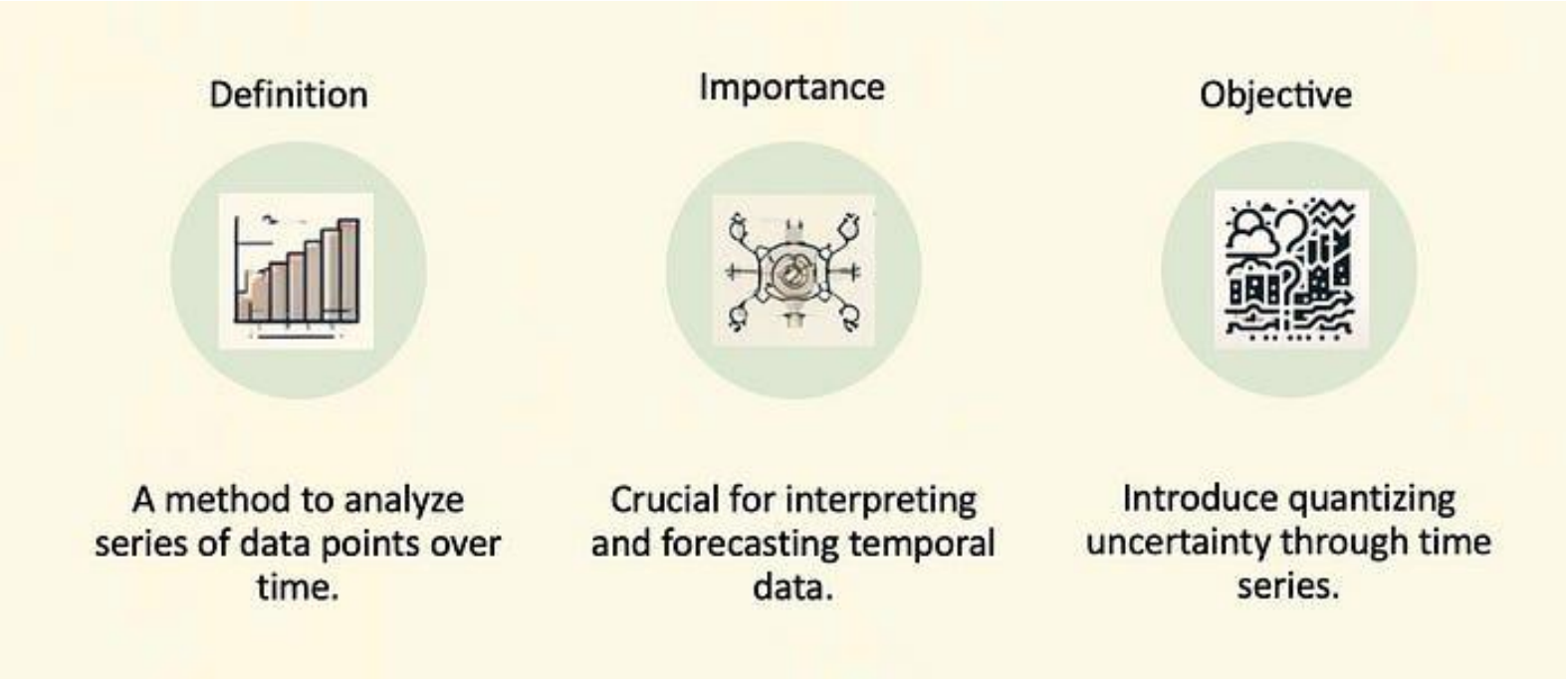
AI VIETNAM
All-in-One Course
(TA Session)

Introduction to Medical Time Series Data

TA - Thanh Huy

Definition

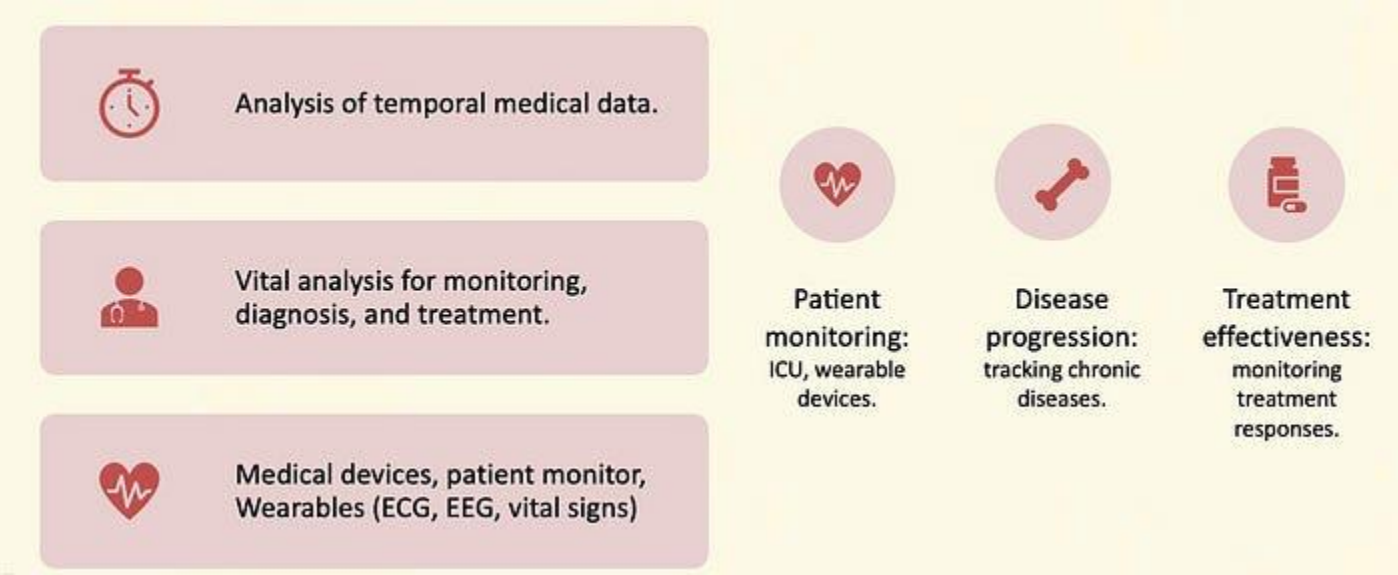
Time series analysis is a statistical technique that analyzes time-ordered data (temporal) points to extract meaningful statistics and other characteristics. It's widely used across various sectors like finance, meteorology, and particularly medical and healthcare.



Medical Time Series Analysis & Applications

Medical time series data, generated from continuous monitoring of patient health metrics, holds immense potential for transforming healthcare. However, leveraging this data effectively involves navigating complex challenges including data variability, privacy concerns, and regulatory compliance.

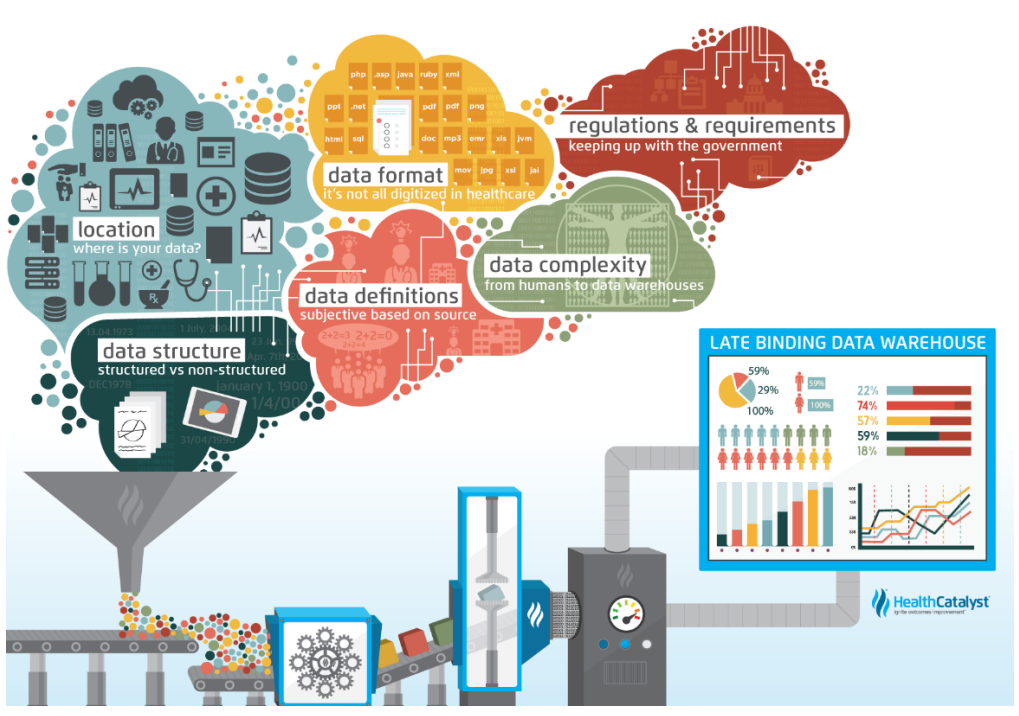
1. **Disease Progression Monitoring:** Continuous monitoring devices using AI can analyze the progression of chronic diseases in real-time, allowing for timely interventions.
2. **Predictive Analytics in Patient Care:** AI models predict potential health events by analyzing patterns from historical data, facilitating preemptive medical actions which can save lives.
3. **Personalized Medicine:** AI-driven analysis helps in tailoring treatment plans to individual patient profiles based on their unique time series data.



Challenges in Medical Time Series

1) Data Complexity and Variability: is derived from a global population that exceeds 8 billion individuals, each with unique physiological and medical profiles. This variability introduces significant complexity in data analysis:

- **Heterogeneity**: Data collected from different individuals under various conditions can be highly heterogeneous, complicating the task of creating uniform models.
- **Scale**: The sheer volume of data generated continuously from health monitoring devices challenges existing computational resources and analytical methods.



Challenges in Medical Time Series

2) Privacy Concerns: The sensitivity of personal health data mandates stringent privacy protections, which are legally enforced by regulations like the EU AI Act [1]. These concerns present several challenges:

- **Data Security**: Ensuring that all data storage and processing mechanisms are secure against breaches and unauthorized access is crucial and often requires sophisticated encryption and access control measures.
- **Compliance**: Adhering to international privacy laws such as GDPR in Europe, which requires that data not only be protected but also that patients have rights over their data, including the right to be forgotten.

Challenges in Medical Time Series

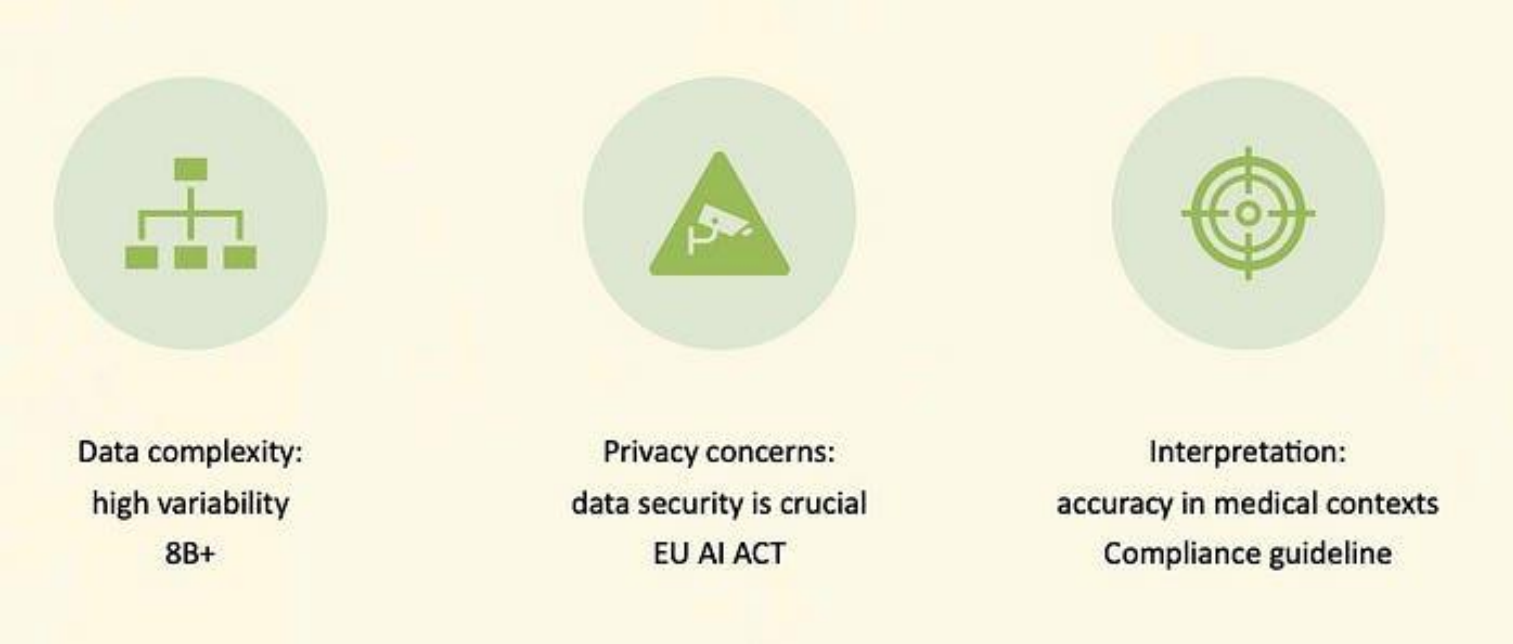
3) Regulatory Compliance: The application of AI in healthcare, particularly in diagnostics, is heavily regulated. Ensuring that AI systems comply with these regulations is a significant challenge:

- **Standards Alignment**: AI models must align with the high standards of accuracy, reliability, and safety set by regulatory bodies such as the FDA in the USA and Health Canada. This often requires extensive validation and testing phases, slowing down the implementation process.
- **Interpretability**: AI systems must not only be effective but also interpretable, meaning that healthcare providers must be able to understand and justify the AI's decision-making process, particularly in critical care scenarios.

Challenges in Medical Time Series

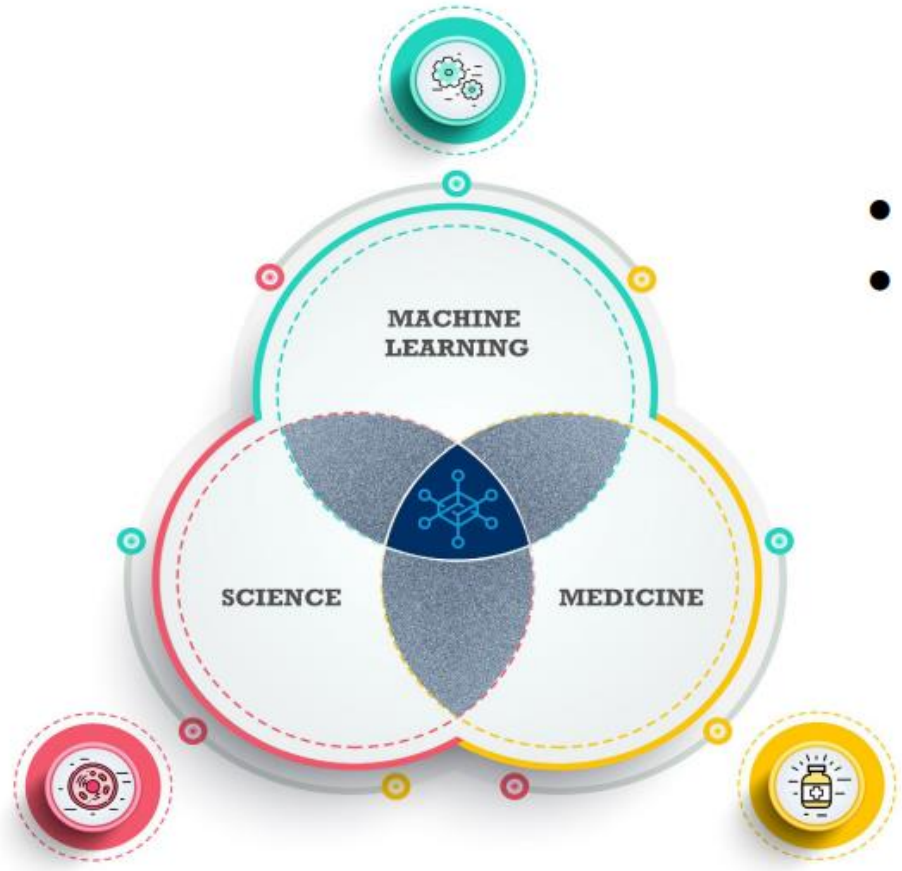
4) Interpretation of Results: The interpretation of AI-generated results from medical time series data must be precise and reliable, as incorrect interpretations can lead to misdiagnoses and inappropriate treatments:

- **Diagnostic Accuracy**: Ensuring that AI tools provide accurate diagnoses that match or exceed the standards of human healthcare providers remains a challenge.
- **Integration into Clinical Workflows**: Even when AI tools are accurate, integrating their use into existing clinical workflows without disrupting standard care practices requires careful planning and training.



Machine Learning x Biomedical

- **MACHINE LEARNING & MEDICINE/HEALTHCARE/BIO-SCIENCE**

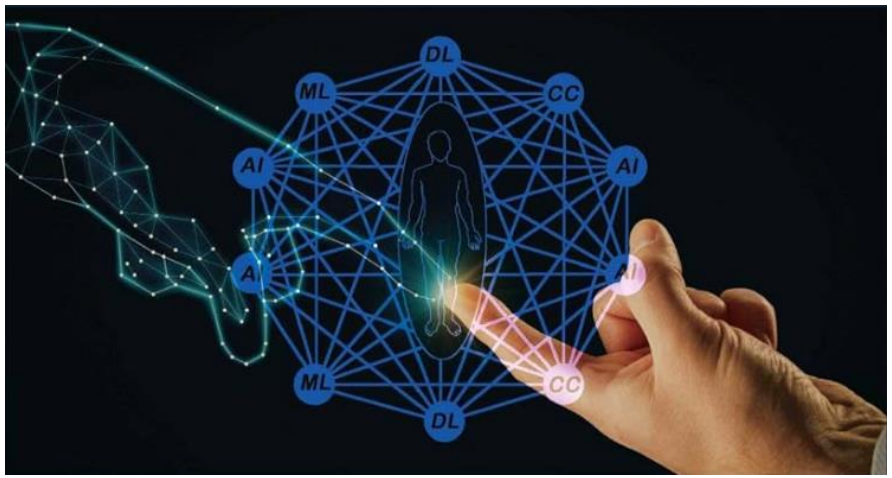


- ML/AI drives a revolution in medicine
- Medicine drives innovations in ML/AI

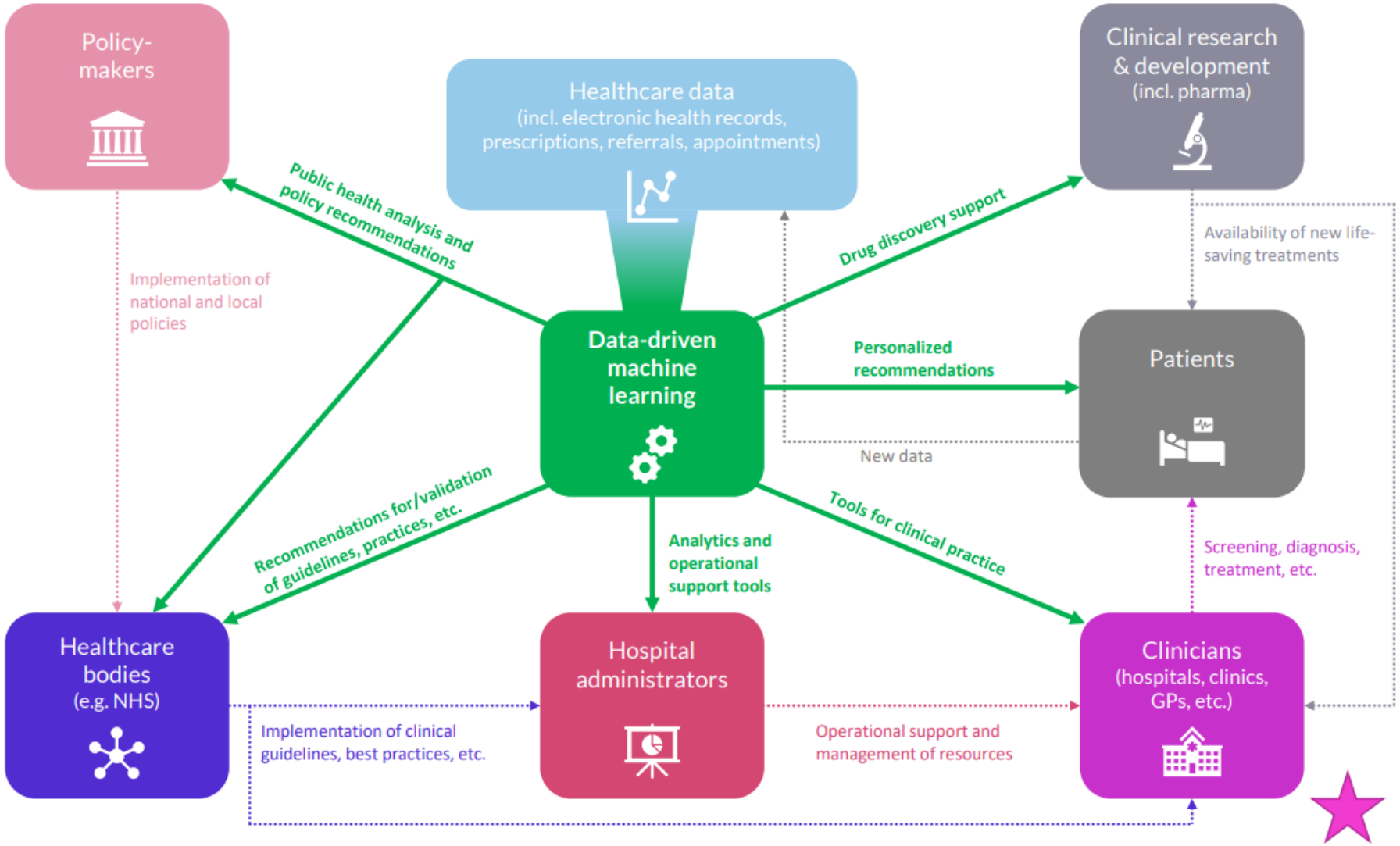
Machine Learning x Biomedical Science

❑ MACHINE LEARNING CAN TRANSFORM MEDICINE & HEALTHCARE

1. deliver precision medicine at the patient level
2. understand the basis and trajectories of health and disease
3. empower healthcare professionals and patients
4. inform and improve clinical pathways, better utilize resources & reduce costs
5. transform population health and public health policy
6. enable new discoveries – clinical, therapeutics



Machine Learning x Biomedical Science



The Usage of ML in Biomedical Science

- THE "AUGMENTED" CLINICIAN, RESEARCHER, PATIENT

Machine learning

…can't do medicine!

…can provide interpretable, trustworthy, actionable information!

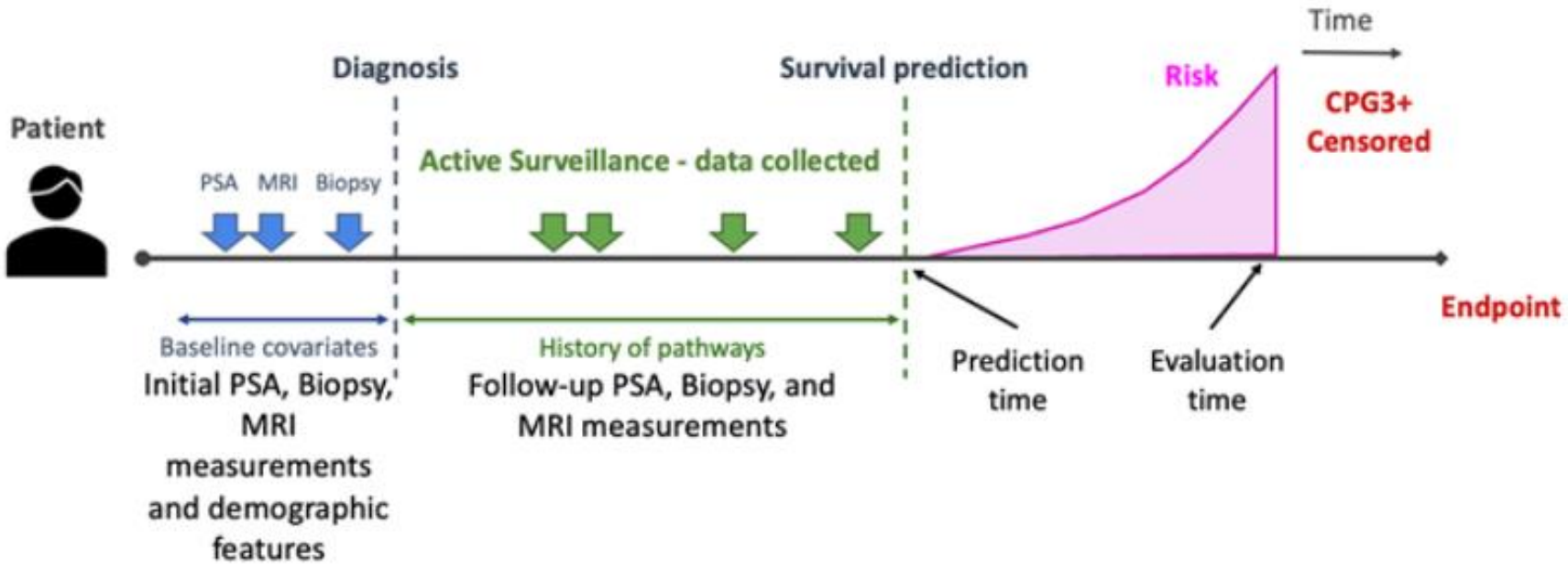
Integrated Clinical Decision Ecosystem using ML

- An integrated clinical decision support ecosystem using machine learning to provide patient-level recommendations and support

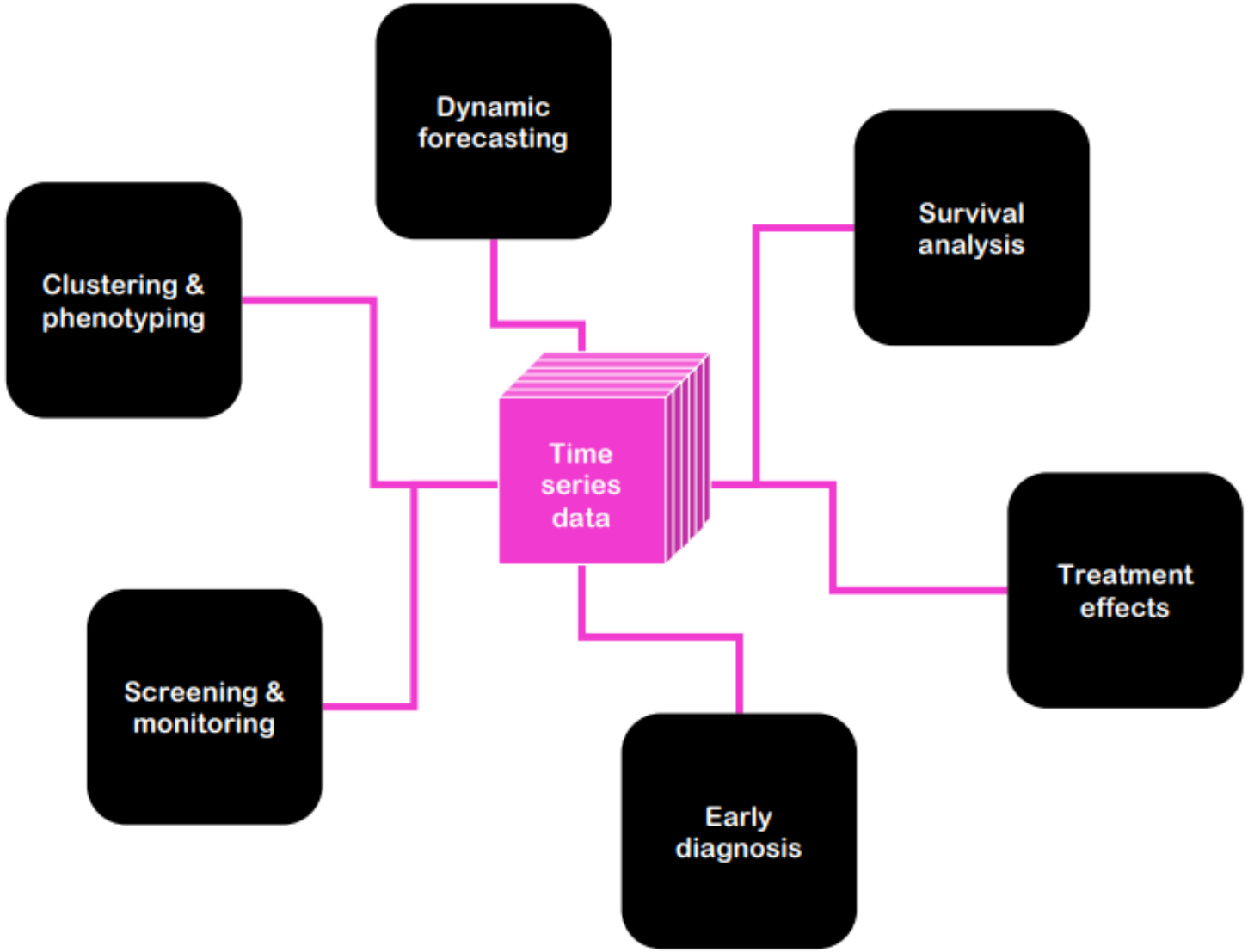
Integrated care:
- Prevention
- Screening
- (Early) Diagnosis
- Treatment
- Monitoring

Multiple venues/areas:
- In-patient/out-patient
- At home

Many stakeholders in every stage of care
- Clinicians, nurses
- Healthcare planners
- Patients!



Time-series in Healthcare: Multi-faceted Problem



Time-series in Healthcare: Multi-faceted Problem

1) Dynamic forecasting
2) Time-to-event and survival analysis
3) Clustering and phenotyping
4) Screening and monitoring
5) Early diagnosis
6) Treatment effects
7) AutoML
8) Interpretability
9) Uncertainty estimation
10) Missing data and informatively missing data
11) Synthetic data generation

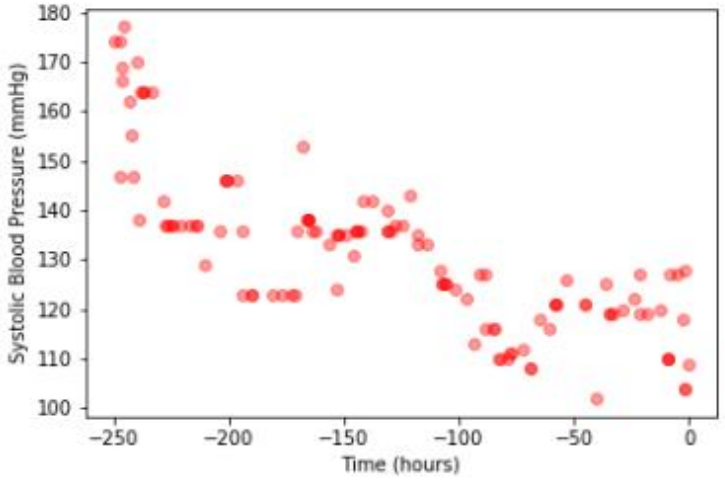
- Reproducibility and visualization

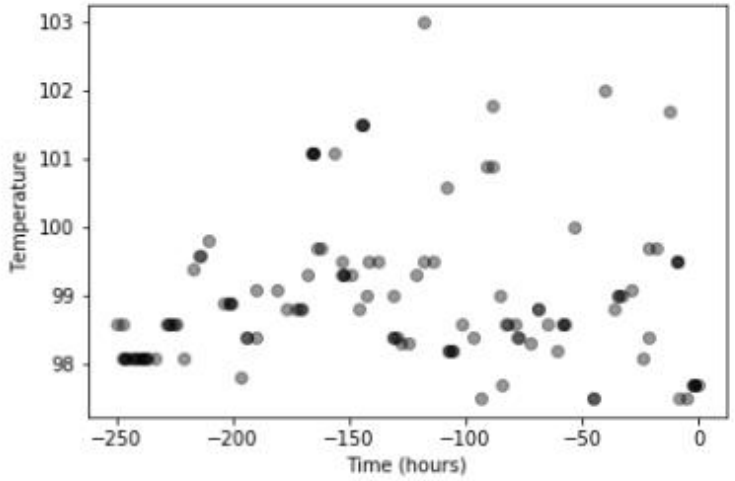
Part 1: tailoring development of time series models to healthcare challenges

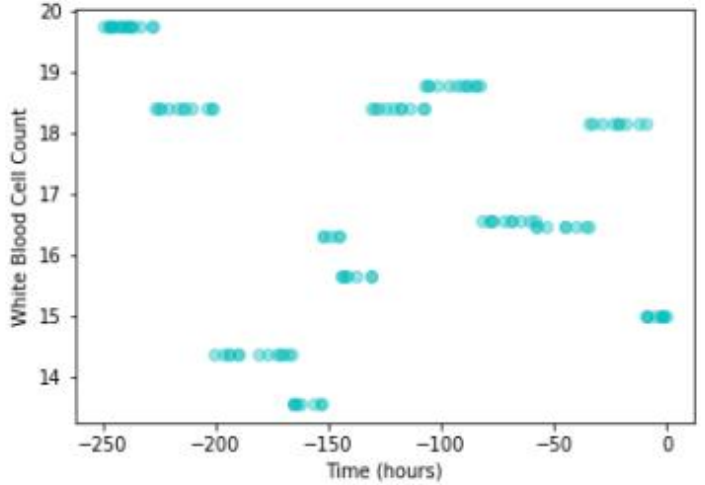
Part 2: making time series models as useful as possible

Healthcare Time Series Data - Unique Challenges

- Multiple streams of measurements
- Measurements are sparse, irregularly and informatively sampled
- Multiple outcomes of interest (various events of interest, various morbidities)
- True clinical states are unobserved (e.g., onset of diseases)
- Many possible patterns (heterogeneous phenotypes, comorbidities)

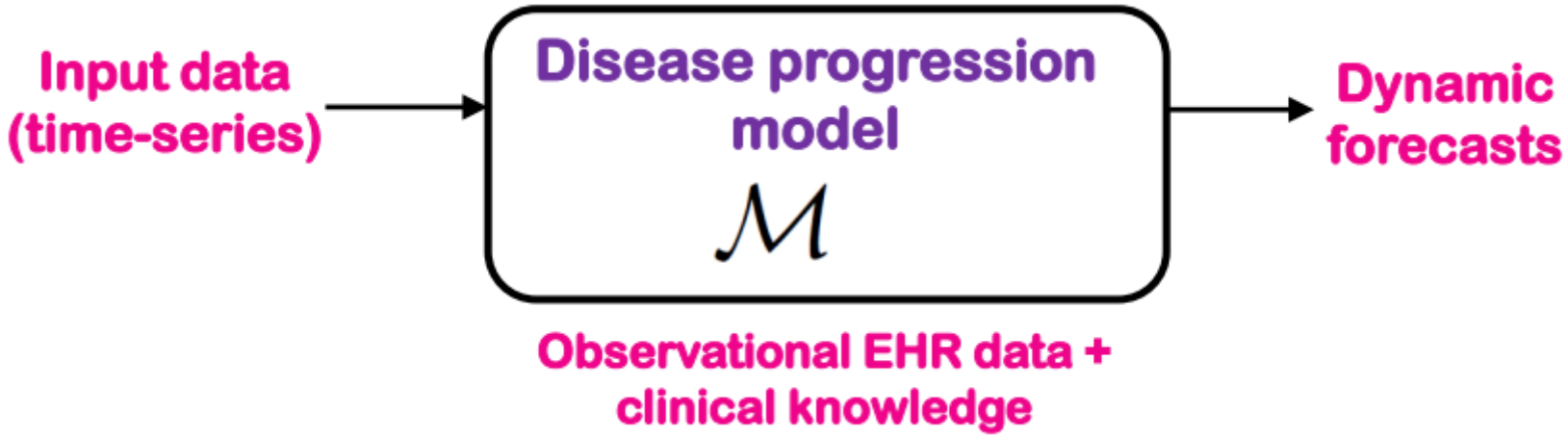






Healthcare Time Series Data - Unique Challenges

- Build disease progression models
  - Understand and model carefully the available data!
- Learn the model parameters from available EHR data (Training time)
- Issue dynamic forecasts for the patient at hand (Test time/Run-time)
- Unravel new understanding of disease progression
  - Population
  - Sub-groups of patients
  - Personalized

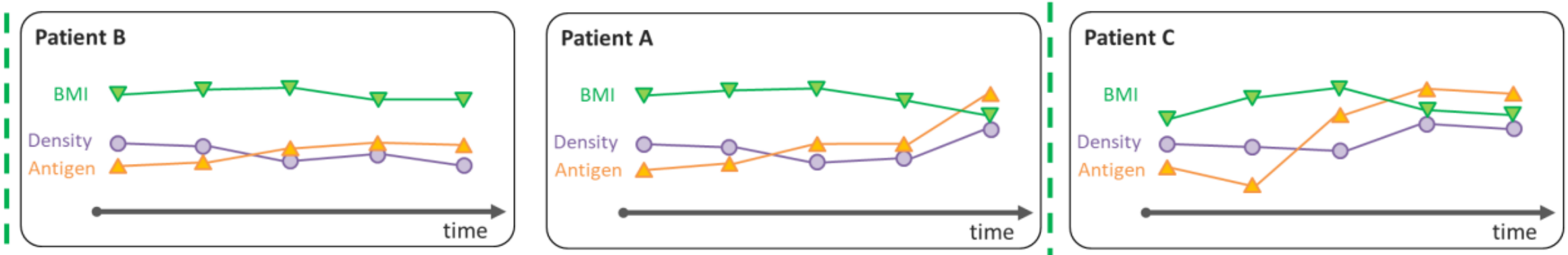


Healthcare Time Series Data - Unique Challenges

- Motivation: How should we group patients?

Example of 3 patients diagnosed with breast cancer (BC)

Should we group patients based on similarity in the time-series observations?



conventional notion of clustering

Key idea: similarity in time-series observations

(e.g. dynamic time warping, auto-encoders)

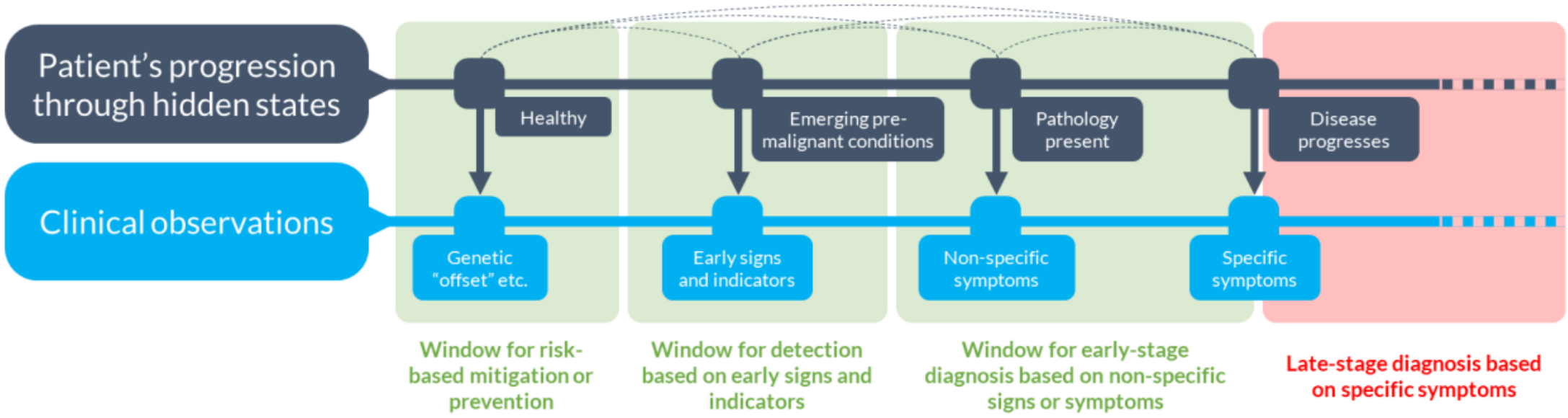
Autoencoder-based approaches

- N. S. Madiraju et al., 2018
- Q. Ma et al., 2019

Healthcare Time Series Data - Unique Challenges

- How can we detect disease early?

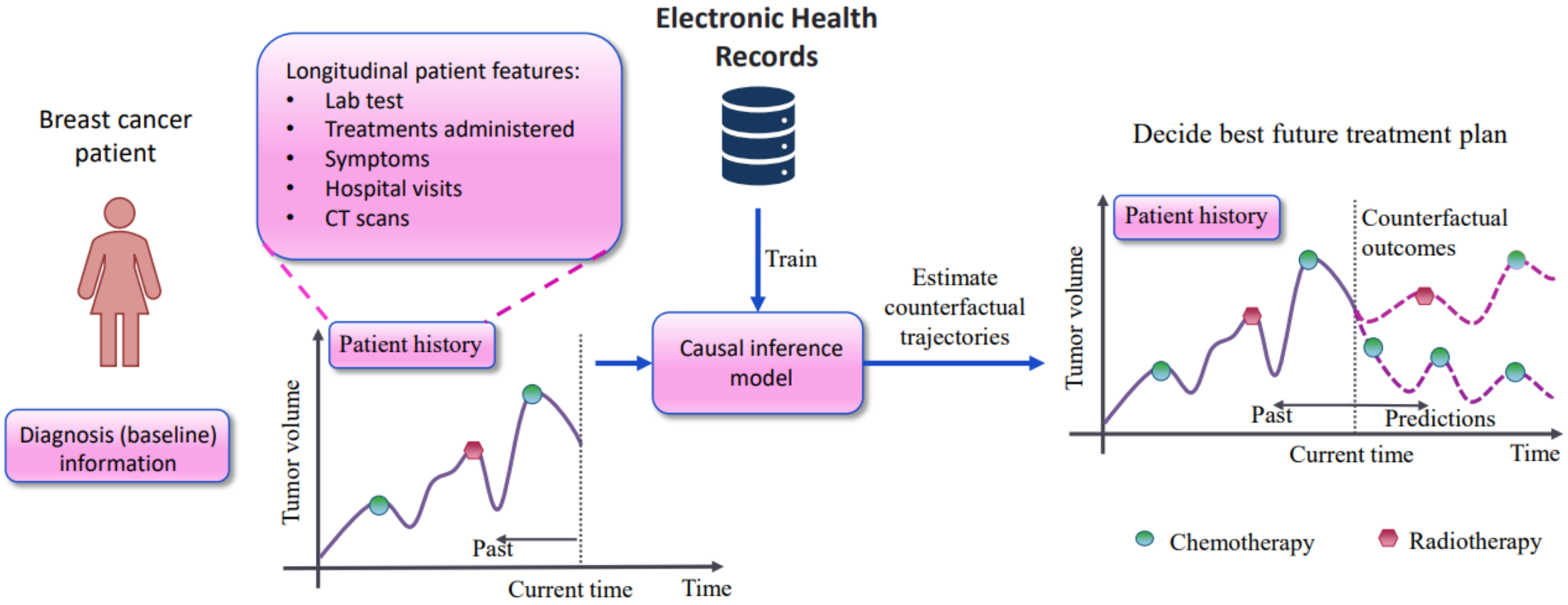
Early diagnosis is more than just event prediction/forecasting
- It involves unravelling and dissecting the underlying states of disease progression towards the event of interest



A quantitative understanding of disease progression is needed!

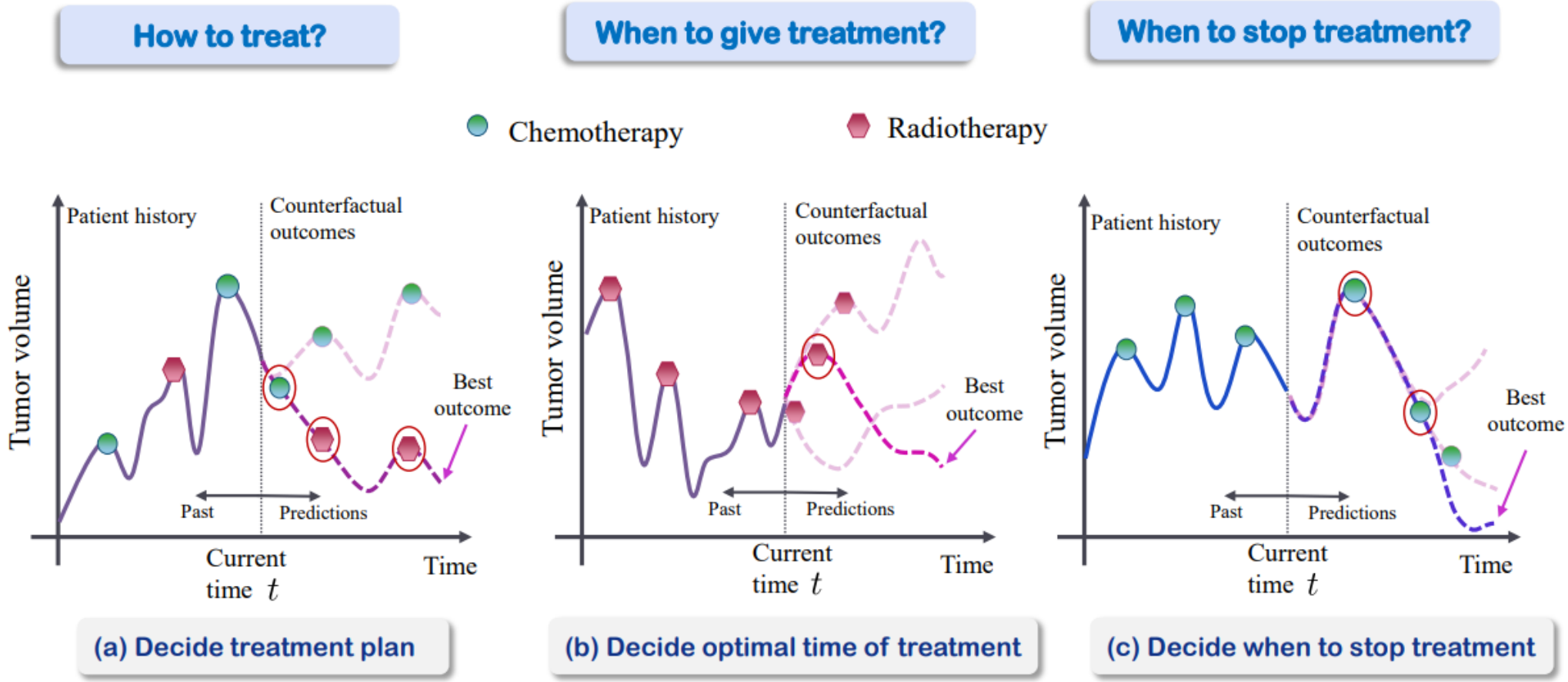
Healthcare Time Series Data - Unique Challenges

- **Individualized treatment effects over time**



Healthcare Time Series Data - Unique Challenges

- Individualized treatment effects over time



Healthcare Time Series Data - Unique Challenges

- Healthcare data: not easy to access

Strict regulations for data access

…the result of perfectly valid concerns regarding privacy

Lack of high-quality healthcare data: impedes ML research in healthcare!

Healthcare Time Series Data - Unique Challenges

- De-identified data vs synthetic data

De-identified/anonymized data: real data with all personal identifiers removed/data fields scrambled

Synthetic data: data created from scratch, cannot be synced back to any individual (if modeled properly)

Requires ML/statistical modelling!

ICML 2021 Tutorial

Generating synthetic data to be used for machine learning modeling is itself a machine learning problem!

Healthcare Time Series Data - Unique Challenges

- Time-series generation

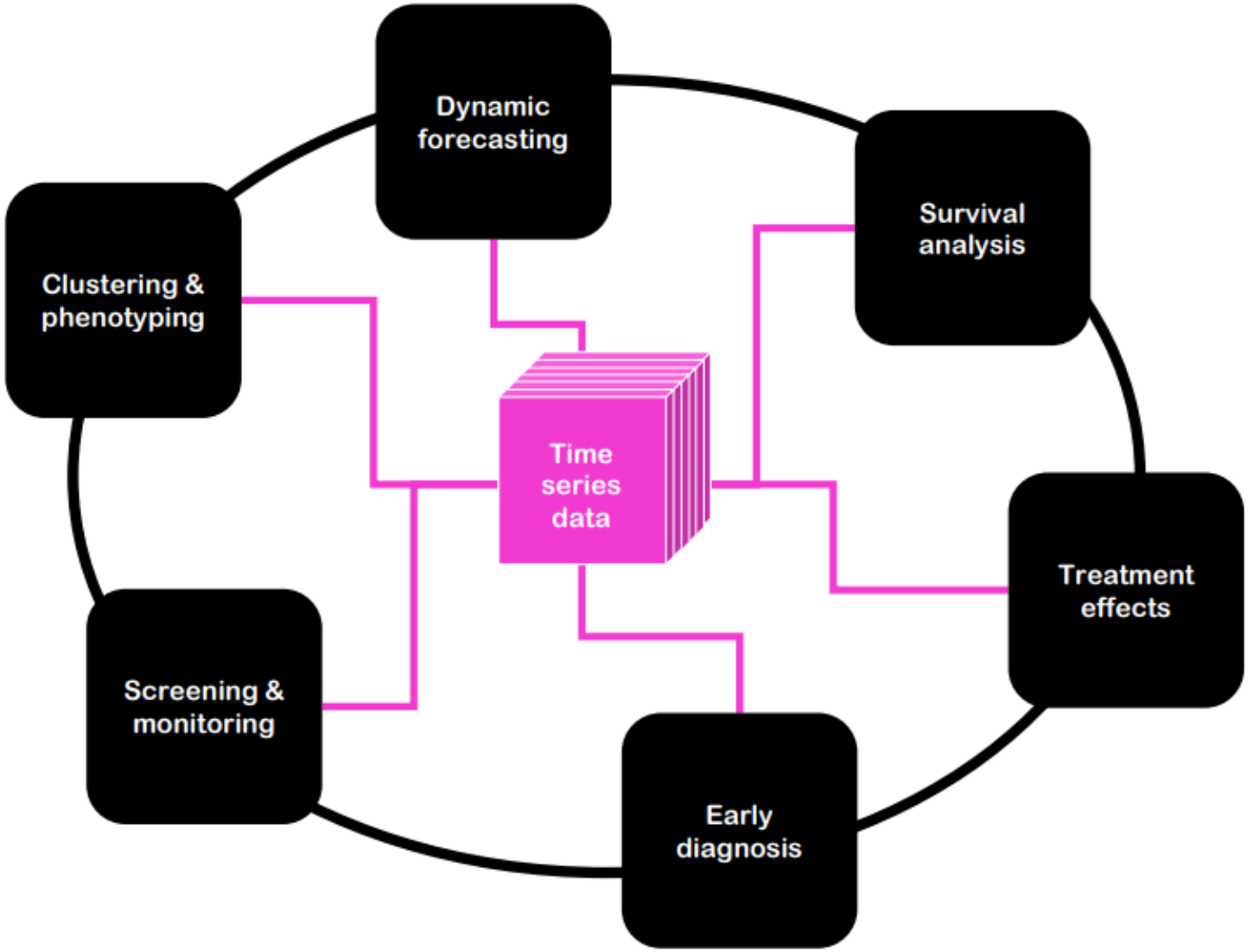
Objective: To generate time-series data with preserving temporal dynamics

Key Example: Synthetic time-series healthcare data generation

Challenges: Capture the distributions of features within each time point as well as complex dynamics of those variables across time points

Conclusion

- NEW FRONTIERS: HEALTHCARE PROBLEMS (AND MODELS) ARE INTERCONNECTED



Conclusion

- **NEW FRONTIERS: HEALTHCARE PROBLEMS (AND MODELS) ARE INTERCONNECTED**

	Patient-oriented	Profession-oriented
Individual	**Bespoke medicine** • Risk scores • Competing risks • Screening and monitoring • Diagnostic support • Longitudinal disease trajectories • Treatment effects	**Empowering healthcare professionals** • Personalised ML assistants to support clinicians • Interpretable, explainable, trustworthy • Multi-disciplinary clinical contributions
At scale	**Population health and public health policy** • Discover & disentangle public risks and risk factors • Population risk assessment → personalized risk • Data-driven guidelines, protocols, standards • Cross-country learning and interventions	**Systems, pathways and processes** • Improving healthcare pathways • Integrating and curating data sources • Integrating a multitude of analytics into delivery systems • Cooperation, interaction and learning

Catalyze a revolution in healthcare

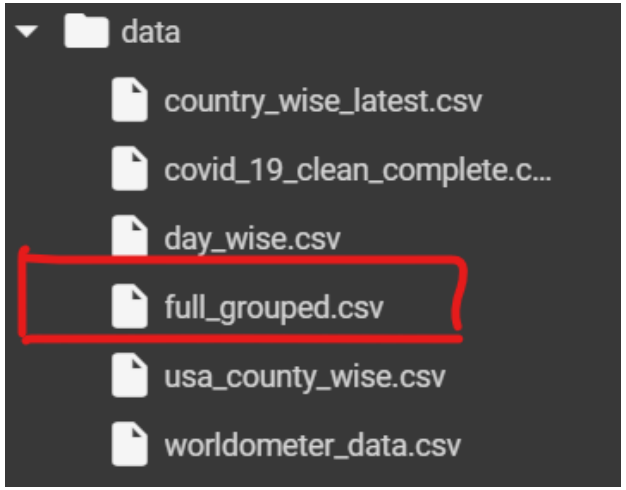
AI VIETNAM
All-in-One Course
(TA Session)

Death Forecasting in Covid-19

TA - Thanh Huy

STA – Hoang Thien

Introduction



	Date	Country/Region	Confirmed	Deaths	Recovered	Active	New cases	New deaths	New recovered	WHO Region
0	2020-01-22	Afghanistan	0	0	0	0	0	0	0	Eastern Mediterranean
1	2020-01-22	Albania	0	0	0	0	0	0	0	Europe
2	2020-01-22	Algeria	0	0	0	0	0	0	0	Africa
3	2020-01-22	Andorra	0	0	0	0	0	0	0	Europe
4	2020-01-22	Angola	0	0	0	0	0	0	0	Africa
5	2020-01-22	Antigua and Barbuda	0	0	0	0	0	0	0	Americas
6	2020-01-22	Argentina	0	0	0	0	0	0	0	Americas
7	2020-01-22	Armenia	0	0	0	0	0	0	0	Europe
8	2020-01-22	Australia	0	0	0	0	0	0	0	Western Pacific
9	2020-01-22	Austria	0	0	0	0	0	0	0	Europe

- **Date:** the date of record
- **Country/Region**: the country/region of record
- **WHO region**: which the WHO region that record located in

Introduction

	Date	Country/Region	Confirmed	Deaths	Recovered	Active	New cases	New deaths	New recovered	WHO Region
0	2020-01-22	Afghanistan	0	0	0	0	0	0	0	Eastern Mediterranean
1	2020-01-22	Albania	0	0	0	0	0	0	0	Europe
2	2020-01-22	Algeria	0	0	0	0	0	0	0	Africa
3	2020-01-22	Andorra	0	0	0	0	0	0	0	Europe
4	2020-01-22	Angola	0	0	0	0	0	0	0	Africa
5	2020-01-22	Antigua and Barbuda	0	0	0	0	0	0	0	Americas
6	2020-01-22	Argentina	0	0	0	0	0	0	0	Americas
7	2020-01-22	Armenia	0	0	0	0	0	0	0	Europe
8	2020-01-22	Australia	0	0	0	0	0	0	0	Western Pacific
9	2020-01-22	Austria	0	0	0	0	0	0	0	Europe

- **Confirmed:** how many covid-diagnosed cases in total
- **Deaths**: number of deaths in the confirmed cases
- **Recovered**: number of recovered people in the confirmed cases
- **Active**: number of people who still positive with covid in the confirmed cases

=> **Comfirmed = Deaths + Recovered + Active**

Introduction

	Date	Country/Region	Confirmed	Deaths	Recovered	Active	New cases	New deaths	New recovered	WHO Region
0	2020-01-22	Afghanistan	0	0	0	0	0	0	0	Eastern Mediterranean
1	2020-01-22	Albania	0	0	0	0	0	0	0	Europe
2	2020-01-22	Algeria	0	0	0	0	0	0	0	Africa
3	2020-01-22	Andorra	0	0	0	0	0	0	0	Europe
4	2020-01-22	Angola	0	0	0	0	0	0	0	Africa
5	2020-01-22	Antigua and Barbuda	0	0	0	0	0	0	0	Americas
6	2020-01-22	Argentina	0	0	0	0	0	0	0	Americas
7	2020-01-22	Armenia	0	0	0	0	0	0	0	Europe
8	2020-01-22	Australia	0	0	0	0	0	0	0	Western Pacific
9	2020-01-22	Austria	0	0	0	0	0	0	0	Europe

- **New cases:** how many new covid-diagnosed cases
- **New deaths**: number of new deaths in the confirmed cases
- **New recovered**: number of new recovered people in the confirmed cases

$\Rightarrow$ **New cases = Confirmed(t) – Confirmed(t – 1)**
New deaths = Deaths(t) – Deaths(t – 1)
New recovered = Recovered(t) – Recovered(t – 1)

Introduction

	Date	Country/Region	Confirmed	Deaths	Recovered	Active	New cases	New deaths	New recovered	WHO Region
0	2020-01-22	Afghanistan	0	0	0	0	0	0	0	Eastern Mediterranean
1	2020-01-22	Albania	0	0	0	0	0	0	0	Europe
2	2020-01-22	Algeria	0	0	0	0	0	0	0	Africa
3	2020-01-22	Andorra	0	0	0	0	0	0	0	Europe
4	2020-01-22	Angola	0	0	0	0	0	0	0	Africa
5	2020-01-22	Antigua and Barbuda	0	0	0	0	0	0	0	Americas
6	2020-01-22	Argentina	0	0	0	0	0	0	0	Americas
7	2020-01-22	Armenia	0	0	0	0	0	0	0	Europe
8	2020-01-22	Australia	0	0	0	0	0	0	0	Western Pacific
9	2020-01-22	Austria	0	0	0	0	0	0	0	Europe

- Number of records: 35156 samples
- Number of country/regions: 187 samples
- Date: 22/01/2020 – 27/07/2020

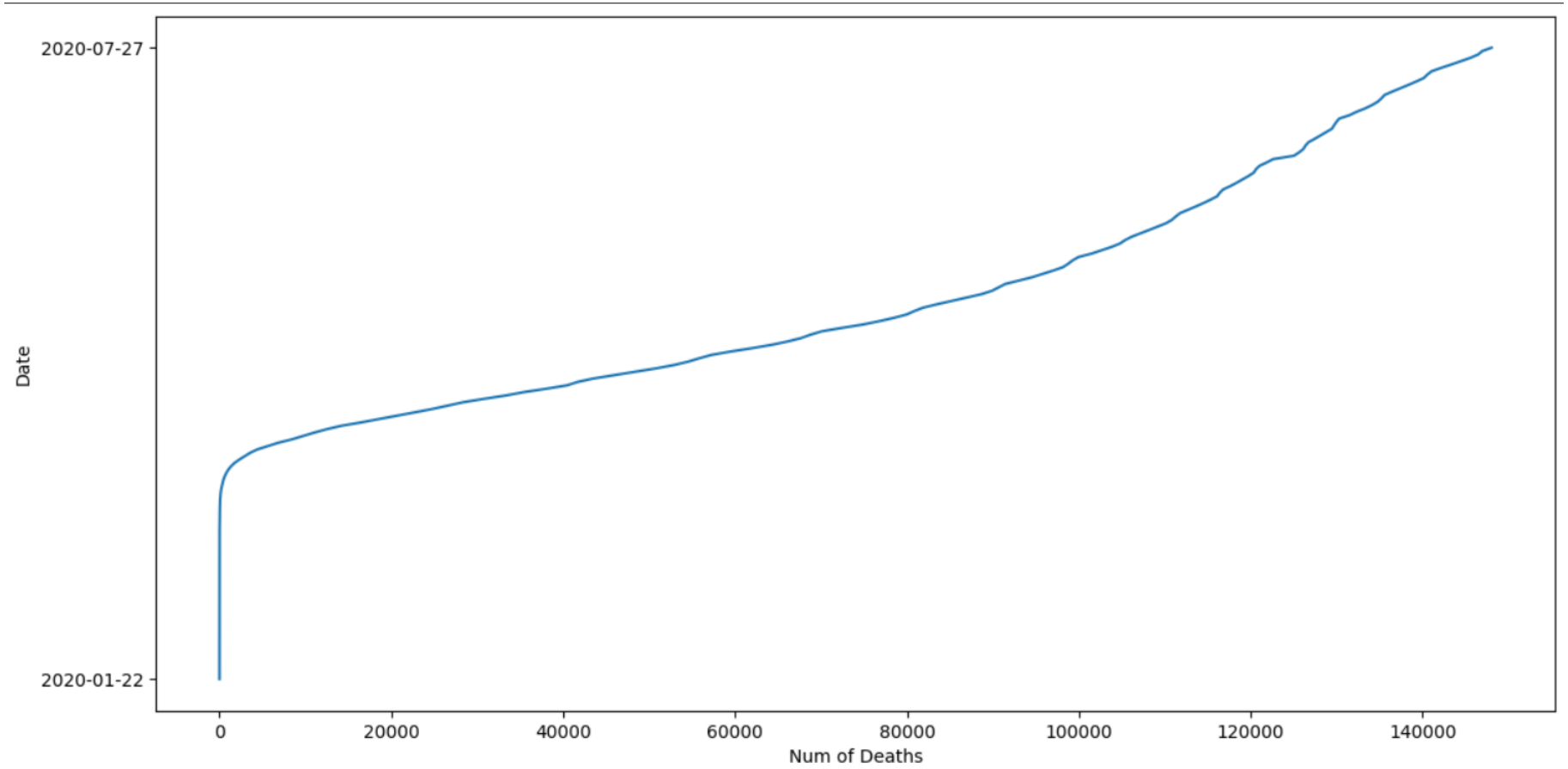
Introduction

	Date	Country/Region	Confirmed	Deaths	Recovered	Active	New cases	New deaths	New recovered	WHO Region
173	2020-01-22	US	1	0	0	1	0	0	0	Americas
360	2020-01-23	US	1	0	0	1	0	0	0	Americas
547	2020-01-24	US	2	0	0	2	1	0	0	Americas
734	2020-01-25	US	2	0	0	2	0	0	0	Americas
921	2020-01-26	US	5	0	0	5	3	0	0	Americas
...	...	...	...	...	...	...	...	...	...	...
34394	2020-07-23	US	4038816	144430	1233269	2661117	68695	1114	22420	Americas
34581	2020-07-24	US	4112531	145560	1261624	2705347	73715	1130	28355	Americas
34768	2020-07-25	US	4178970	146465	1279414	2753091	66439	905	17790	Americas
34955	2020-07-26	US	4233923	146935	1297863	2789125	54953	470	18449	Americas
35142	2020-07-27	US	4290259	148011	1325804	2816444	56336	1076	27941	Americas

188 rows × 10 columns

- Number of records: 188 samples
- Date: 22/01/2020 – 27/07/2020

Introduction



- Number of records: 188 samples
- Date: 22/01/2020 – 27/07/2020

Preprocessing

- Convert tabular into the dataset including n continuous records and the output as the n + 1 record


```python
# Create sequences of seven consecutive values (X) and the next value (y)
def create_sequences(data, seq_length):
    X, y = [], []
    for i in range(len(data) - seq_length):
        X.append(data[i:i+seq_length])
        y.append(data[i+seq_length])
    return np.array(X), np.array(y)

seq_length = 7
X, y = create_sequences(data, seq_length)
```

Preprocessing

- Split dataset into train set and test set

```
# Split the data into training, validation, and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42, shuffle=False)
X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

```
((126, 7), (126,), (55, 7), (55,))
```

Training

- Initialize the model
- Fit the train set into the model

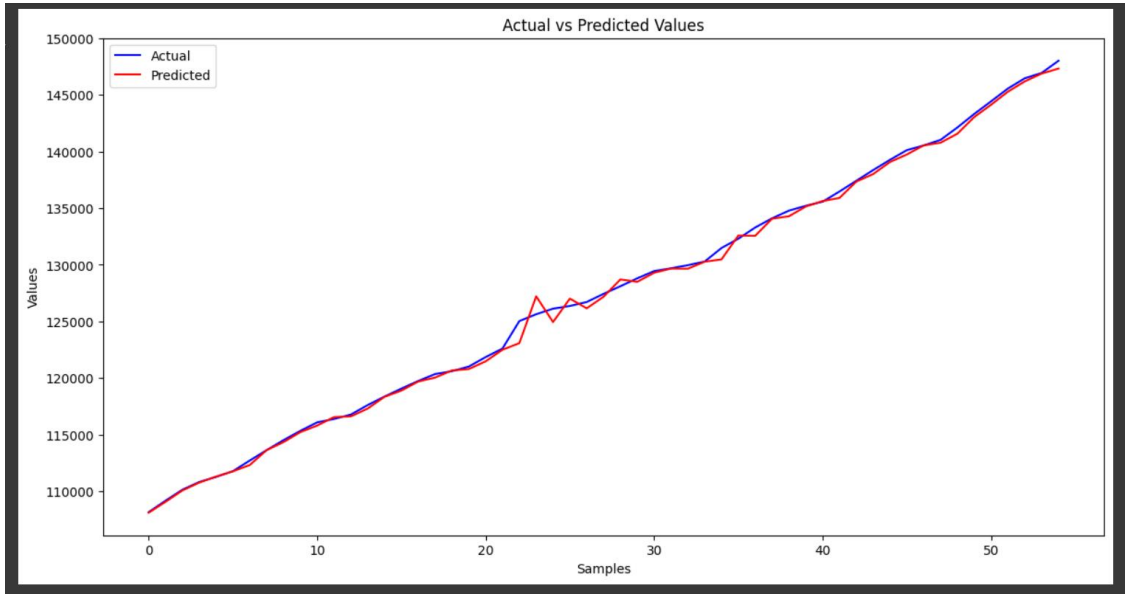
```
[ ]  # Create a linear regression model
     model = LinearRegression()

     # Train the model
     model.fit(X_train, y_train)
```

Testing

- Predict the test set and evaluate using metrics

```python
# Test the model
y_test_pred = model.predict(X_test)
test_mse = mean_squared_error(y_test, y_test_pred)
test_mae = mean_absolute_error(y_test, y_test_pred)
test_r2 = r2_score(y_test, y_test_pred)
print(f'Test MSE: {test_mse}')
print(f'Test MAE: {test_mae}')
print(f'Test R²: {test_r2}')

# Print some predictions
print(f'Predictions: {y_test_pred[:10]}')
print(f'Actual values: {y_test[:10]}')
```

```
Test MSE: 246064.28028447783
Test MAE: 322.5146545441239
Test R²: 0.998044125890038
Predictions: [108110.90112369 109066.31267272 110069.76535847 110766.74237685
 111295.19854804 111761.67833179 112317.87923343 113629.67265811
 114338.91604917 115220.36176308]
Actual values: [108159 109168 110138 110818 111269 111774 112714 113631 114512 115334]
```



AI VIETNAM
All-in-One Course
(TA Session)

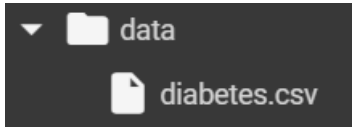
Diabete Classification

TA - Thanh Huy

STA – Hoang Thien

Introduction

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
5	5	116	74	0	0	25.6	0.201	30	0
6	3	78	50	32	88	31.0	0.248	26	1
7	10	115	0	0	0	35.3	0.134	29	0
8	2	197	70	45	543	30.5	0.158	53	1
9	8	125	96	0	0	0.0	0.232	54	1



- **Pregnancies:** To express the Number of pregnancies
- **Glucose:** To express the Glucose level in blood
- **BloodPressure:** To express the Blood pressure measurement

Introduction

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
5	5	116	74	0	0	25.6	0.201	30	0
6	3	78	50	32	88	31.0	0.248	26	1
7	10	115	0	0	0	35.3	0.134	29	0
8	2	197	70	45	543	30.5	0.158	53	1
9	8	125	96	0	0	0.0	0.232	54	1

- **SkinThickness:** To express the thickness of the skin
- **Insulin:** To express the Insulin level in blood
- **BMI:** To express the Body mass index

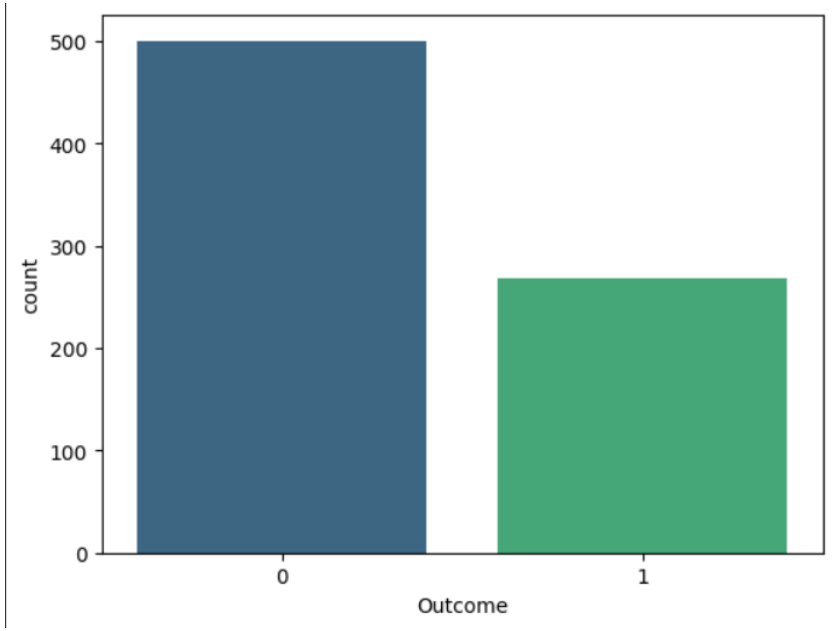
Introduction

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
5	5	116	74	0	0	25.6	0.201	30	0
6	3	78	50	32	88	31.0	0.248	26	1
7	10	115	0	0	0	35.3	0.134	29	0
8	2	197	70	45	543	30.5	0.158	53	1
9	8	125	96	0	0	0.0	0.232	54	1

- **DiabetesPedigreeFunction:** To express the Diabetes percentage
- **Age:** To express the age
- **Outcome:** To express the final result 1 is Yes and 0 is No

Introduction

```
def display_normal_barchart(dataframe, col_name):
    sns.countplot(x=dataframe[col_name], palette='viridis')
    plt.show(block=True)
```



Outcome:

- 0: 500 samples
- 1: 268 samples

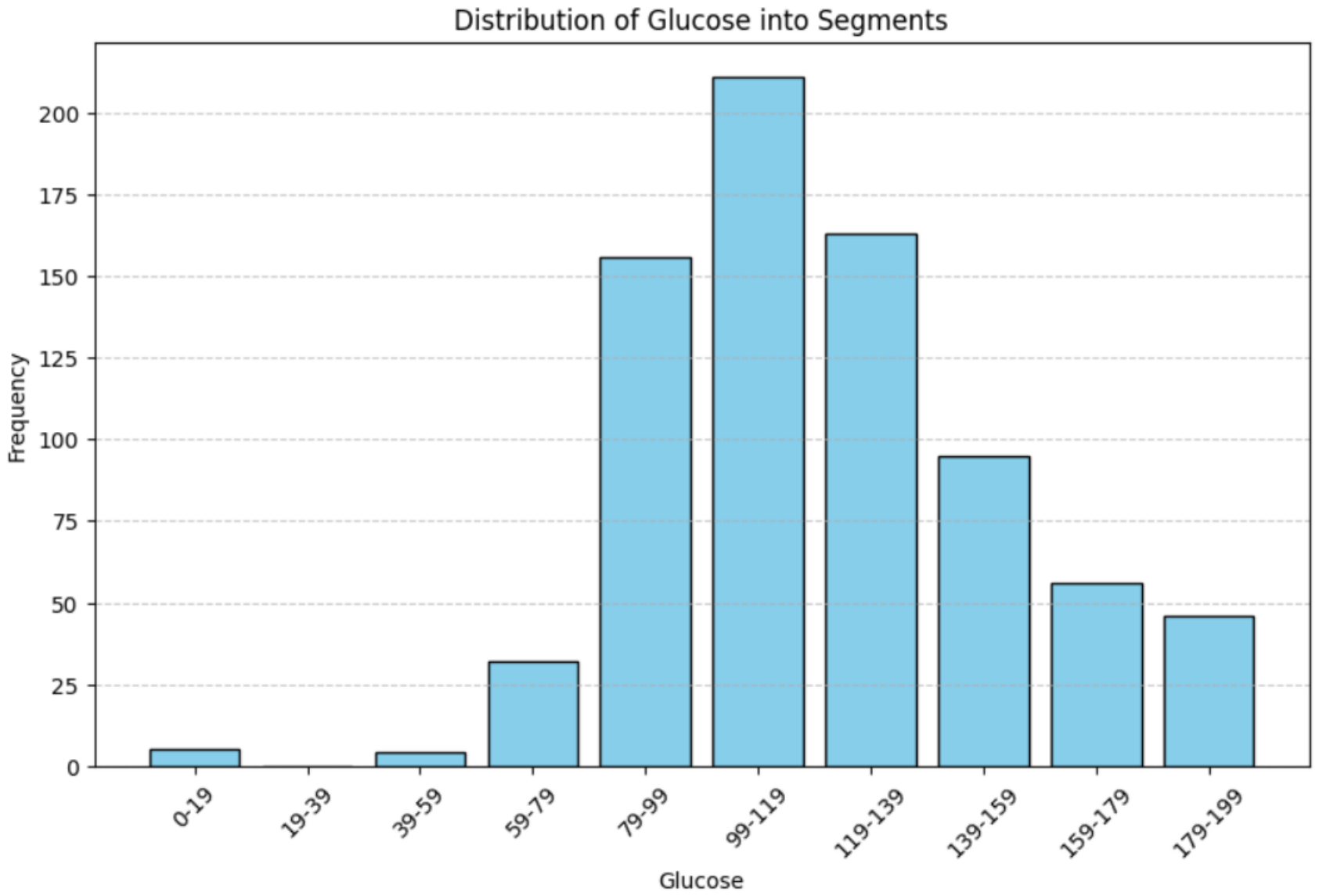
Function

```
def display_barchart_byRatio(dataframe, col_name, split=10):

    # Get Max, Min value
    df = dataframe[col_name]
    min_value = df.min()
    max_value = df.max()

    bins = np.linspace(min_value, max_value, split + 1)  # 11 points create 10 equal bins
    labels = [f'{int(bins[i])}-{int(bins[i+1])}' for i in range(len(bins)-1)]

    # Assign numbers to bins
    df['Segment'] = pd.cut(df, bins=bins, labels=labels, include_lowest=True)

    # Count the frequency of each segment
    segment_counts = df['Segment'].value_counts().sort_index()

    # Plot the bar chart
    plt.figure(figsize=(10, 6))
    plt.bar(segment_counts.index, segment_counts.values, width=0.8, color='skyblue', edgecolor='black')
```

Function

```
# Plot the bar chart
plt.figure(figsize=(10, 6))
plt.bar(segment_counts.index, segment_counts.values, width=0.8, color='skyblue', edgecolor='black')

# Set the labels and title
plt.xlabel(col_name)
plt.ylabel('Frequency')
plt.title(f'Distribution of {col_name} into Segments')

# Rotate the x-tick labels for better readability
plt.xticks(rotation=45)

# Add grid lines for better readability
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Show the plot
plt.show(block=True)
```

Introduction



Checking values

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
5	5	116	74	0	0	25.6	0.201	30	0
6	3	78	50	32	88	31.0	0.248	26	1
7	10	115	0	0	0	35.3	0.134	29	0
8	2	197	70	45	543	30.5	0.158	53	1
9	8	125	96	0	0	0.0	0.232	54	1

Checking values

```
df.isnull().sum()
```

	0
Pregnancies	0
Glucose	0
BloodPressure	0
SkinThickness	0
Insulin	0
BMI	0
DiabetesPedigreeFunction	0
Age	0
Outcome	0

dtype: int64

```
zero_columns = [col for col in df.columns if (df[col].min() == 0 and col not in["Outcome", "Pregnancies"])]
for col in zero_columns:
    df[col] = np.where(df[col] == 0, np.nan, df[col])
```

```
[ ] df.isnull().sum()
```

	0
Pregnancies	0
Glucose	5
BloodPressure	35
SkinThickness	227
Insulin	374
BMI	11
DiabetesPedigreeFunction	0
Age	0
Outcome	0

dtype: int64

Checking values

```
[11] def fill_na_with_median(dataframe):
         dataframe = dataframe.apply(lambda x: x.fillna(x.median()) if x.dtype not in ["category", "object", "bool"] else x, axis=0)
         return dataframe

     df = fill_na_with_median(df)
```

```
df.head(10)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148.0	72.0	35.0	125.0	33.6	0.627	50	1
1	1	85.0	66.0	29.0	125.0	26.6	0.351	31	0
2	8	183.0	64.0	29.0	125.0	23.3	0.672	32	1
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1
5	5	116.0	74.0	29.0	125.0	25.6	0.201	30	0
6	3	78.0	50.0	32.0	88.0	31.0	0.248	26	1
7	10	115.0	72.0	29.0	125.0	35.3	0.134	29	0
8	2	197.0	70.0	45.0	543.0	30.5	0.158	53	1
9	8	125.0	96.0	29.0	125.0	32.3	0.232	54	1

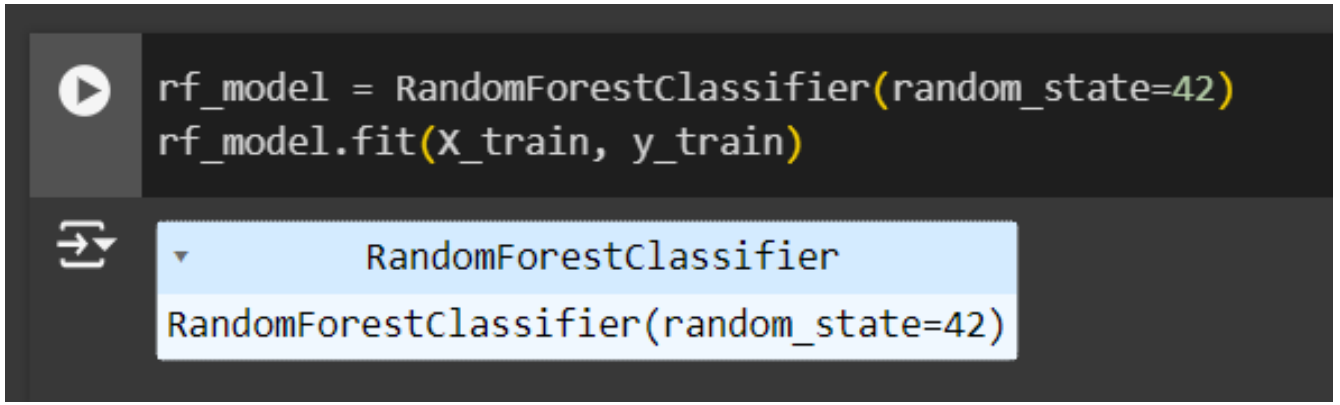
Preprocessing

- Convert data into the attributes(X) and labels(Y - Outcome)
- Split dataset into train set and test set

```
[ ]  target = "Outcome"
     X = df.drop(target, axis=1)
     y = df[target]

     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Training

- Initialize the model
- Fit the train set into the model



Testing

- Predict the test set and evaluate using metrics

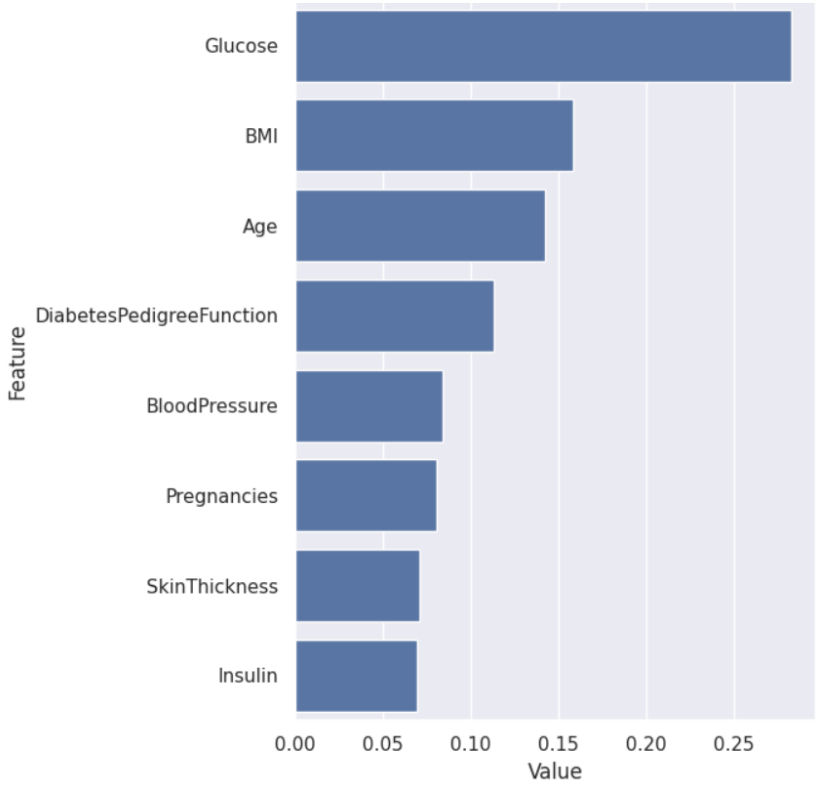
```
[ ]  acc_train = accuracy_score(y_train, rf_model.predict(X_train))
     acc_test = accuracy_score(y_test, rf_model.predict(X_test))

     print(20*"#", "Accuracy & Results", 20*"#")
     print("Accuracy Train : ", "%.3f" % acc_train)
     print("Accuracy Test : ", "%.3f" % acc_test)
```

```
#################### Accuracy & Results ####################
Accuracy Train :  1.000
Accuracy Test :  0.753
```

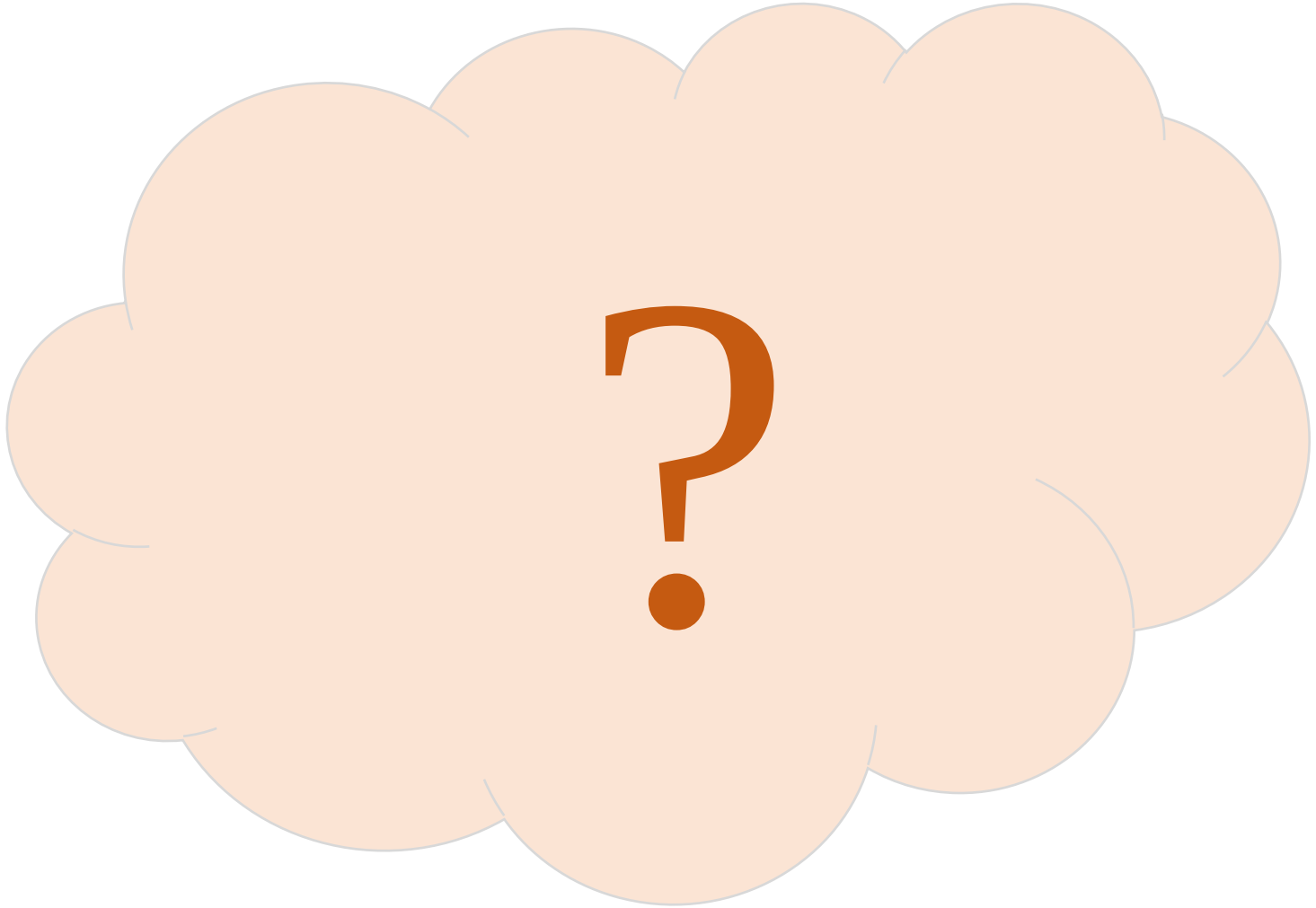
Testing

```
[ ]  feature_imp = pd.DataFrame({'Value' : rf_model.feature_importances_, 'Feature': X.columns})
     plt.figure(figsize=(7,7))
     sns.set(font_scale=1)
     sns.barplot(x='Value', y='Feature', data=feature_imp.sort_values(by='Value', ascending=False))
     plt.title('Feature Importances')
     plt.tight_layout()
     plt.show()
```

- Visualize the weights of attributes of model



Questions




Thank you!