Cosan

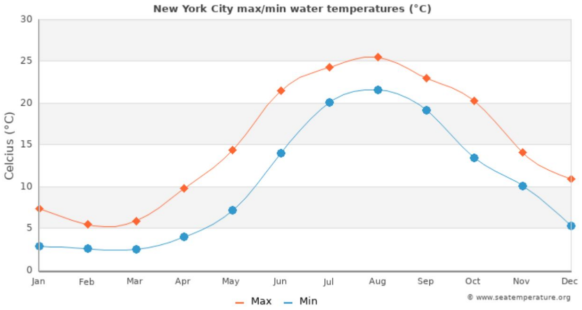
Data analytics library using modern C++

Jiahe Chen (jc5348), Xinyu Zhang (xz2691), Zida Zhou (zz2791)

What is Cosan used for?

Imagine you have n samples data X (input data) and Y (as target data)

- C++ version of data analytics tools that handle preprocessing, model fitting and post processing.



Motivation - why Cosan?

Comparison with current off-the-shelf library:

- Shogun(http://shogun-toolbox.org)
- Scikit-learn(https://scikit-learn.org/stable/)
- MATLAB(https://www.mathworks.com/products/matlab.html)
- R(https://www.r-project.org)

...

Domain knowledge

- Dataset Transformation
  - preprocessing
  - feature generation
  - pipelines
- Linear Models
  - ordinary least squares
  - ridge regression
- Model Selection and Evaluation
  - metrics
  - hyper-parameter tuning
  - cross-validation

Design Goals

1. User-friendly

2. Extensibility & Reusability

3. Portability

4. Utilization of C++ modern features

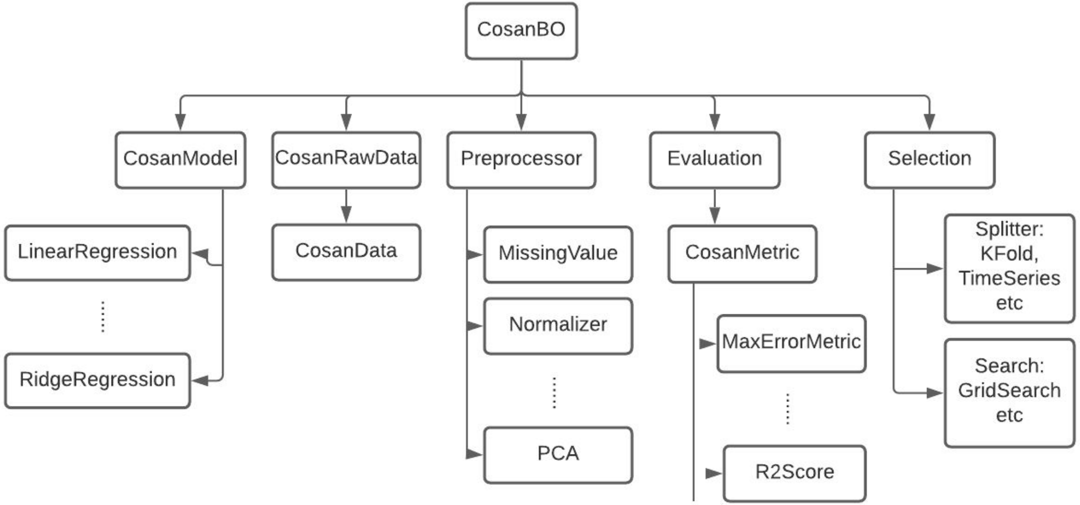
User-friendly: Implementation

1. Applicable to input sources:
   a. csv
   b. direct initialization in data container or from std::vector
2. Customizable data type
   a. CosanData data type: any combo of numeric C++ types
3. Easy to use + tutorial
   a. Tutorial
4. Well-documented
   a. Documentation

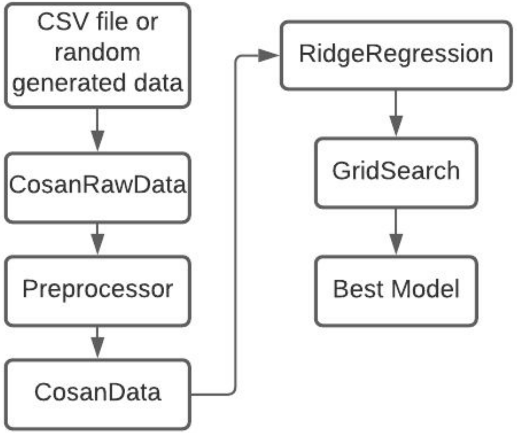
Other Implementation

1. Extensibility & Reusability:
    a. OOP: class hierarchy
2. Portability:
    a. header-only library
3. Utilization of C++ modern features:
    a. Templates, gsl::index, etc

Module Overview:



Typical Workflow:



CosanBO

Modern C++ Features Overview

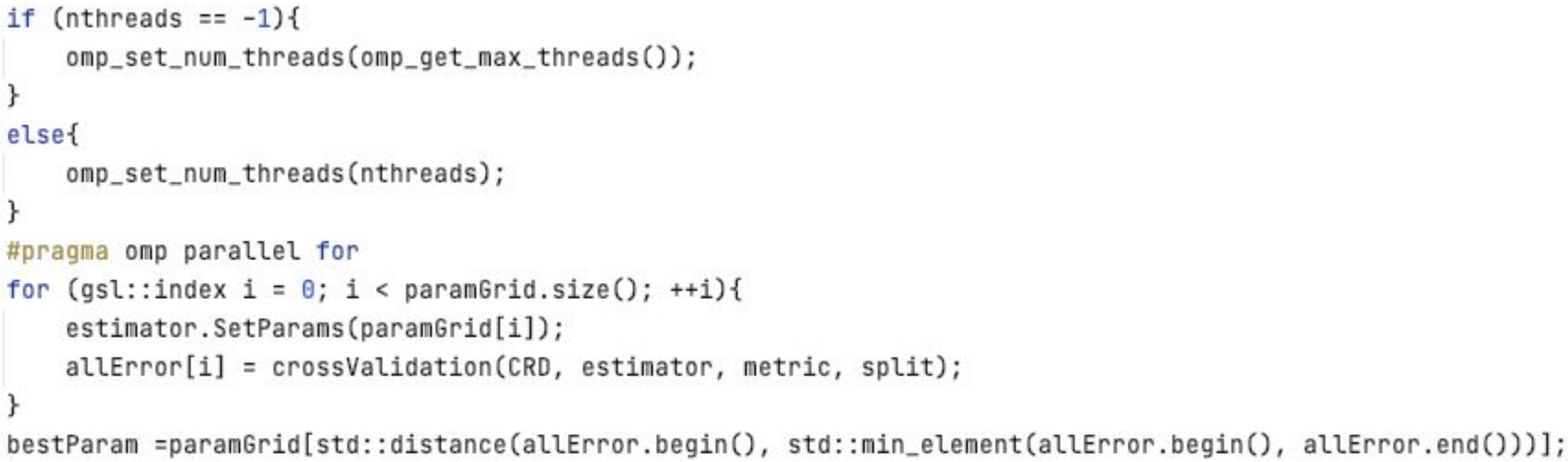
- Concepts
- std::variant
- std::chrono
- Static assertions (static_assert)

...

- Concurrency (OpenMP)
- gsl:: index
- fmt (open-source formatting library)

Concurrency: <omp.h>

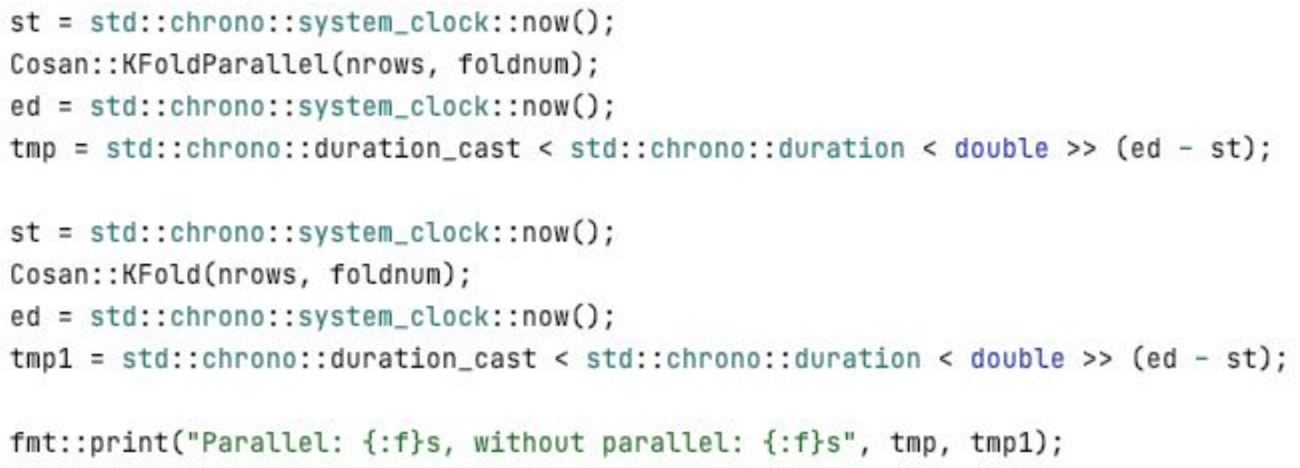
Indexing: gsl::index



```
if (nthreads == -1){
    omp_set_num_threads(omp_get_max_threads());
}
else{
    omp_set_num_threads(nthreads);
}
#pragma omp parallel for
for (gsl::index i = 0; i < paramGrid.size(); ++i){
    estimator.SetParams(paramGrid[i]);
    allError[i] = crossValidation(CRD, estimator, metric, split);
}
bestParam =paramGrid[std::distance(allError.begin(), std::min_element(allError.begin(), allError.end()))];
```

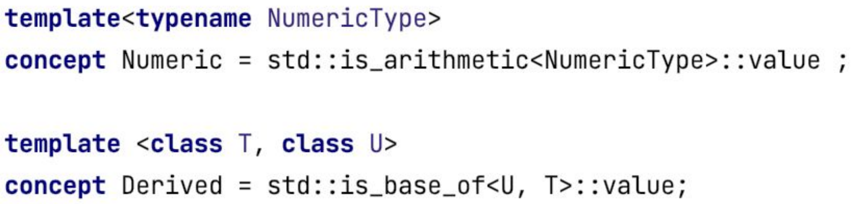
Time, duration, benchmarking: <chrono>

Formatting library, C++20 std::format: <fmt>

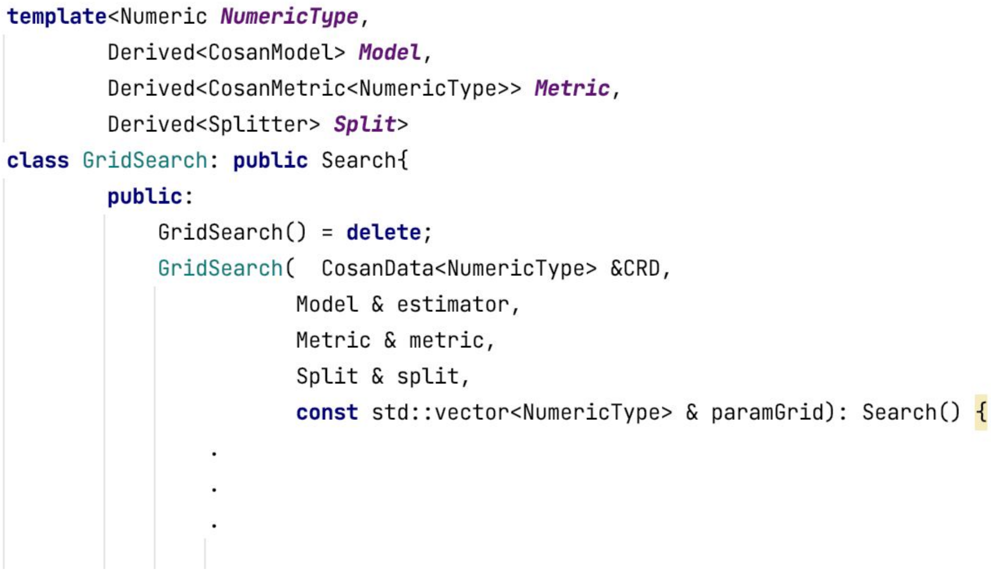


```
st = std::chrono::system_clock::now();
Cosan::KFoldParallel(nrows, foldnum);
ed = std::chrono::system_clock::now();
tmp = std::chrono::duration_cast < std::chrono::duration < double >> (ed - st);

st = std::chrono::system_clock::now();
Cosan::KFold(nrows, foldnum);
ed = std::chrono::system_clock::now();
tmp1 = std::chrono::duration_cast < std::chrono::duration < double >> (ed - st);

fmt::print("Parallel: {:f}s, without parallel: {:f}s", tmp, tmp1);
```

Templates & Concepts for Class

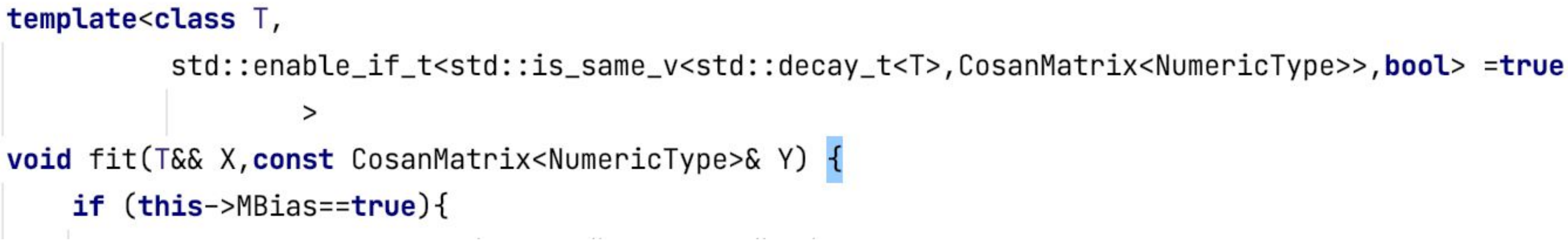


```
template<typename NumericType>
concept Numeric = std::is_arithmetic<NumericType>::value ;

template <class T, class U>
concept Derived = std::is_base_of<U, T>::value;
```



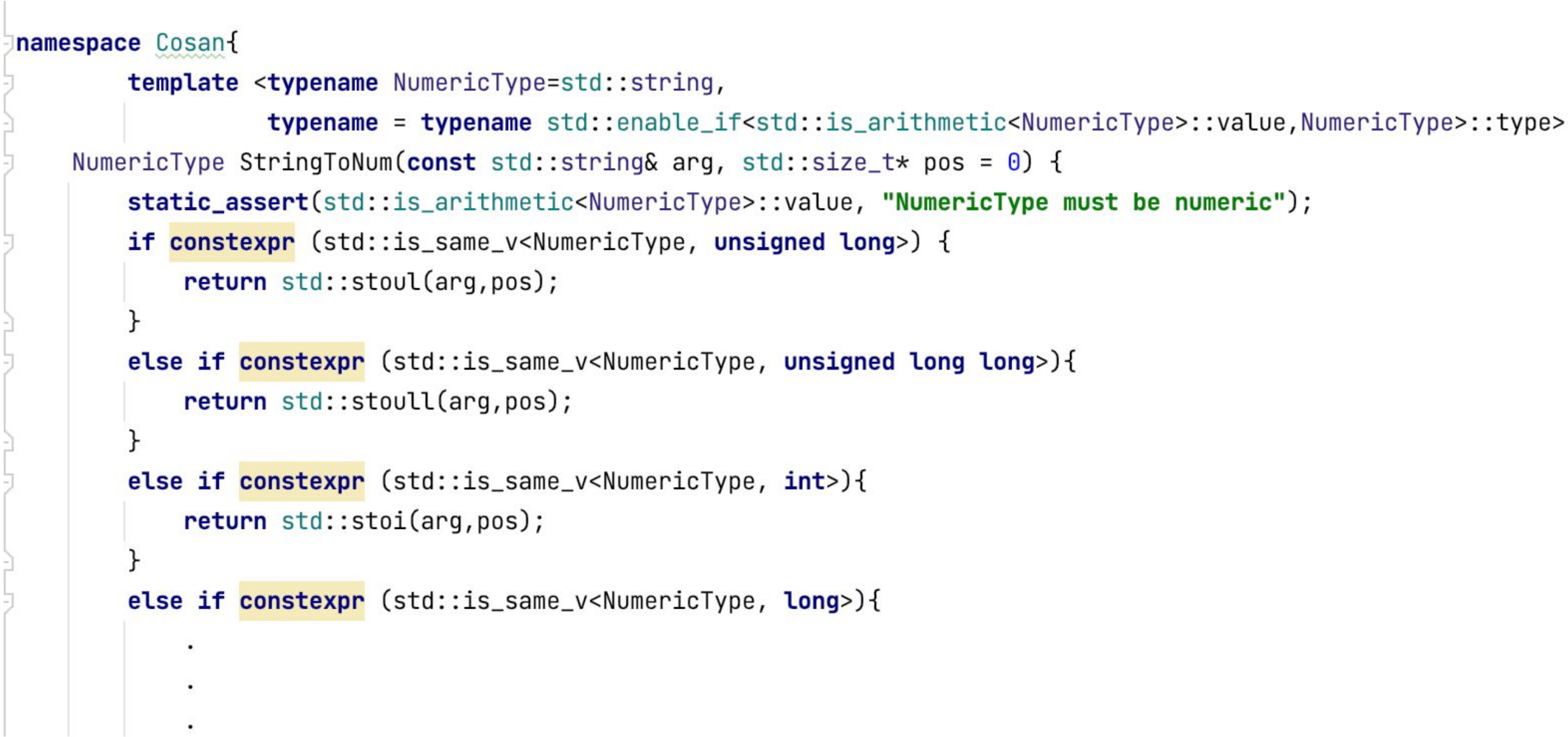
```
template<Numeric NumericType,
        Derived<CosanModel> Model,
        Derived<CosanMetric<NumericType>> Metric,
        Derived<Splitter> Split>
class GridSearch: public Search{
        public:
            GridSearch() = delete;
            GridSearch(  CosanData<NumericType> &CRD,
                         Model & estimator,
                         Metric & metric,
                         Split & split,
                         const std::vector<NumericType> & paramGrid): Search() {
                    .
                    .
                    .
```

Templates for Functions



```cpp
template<class T,
          std::enable_if_t<std::is_same_v<std::decay_t<T>,CosanMatrix<NumericType>>,bool> =true
                  >
void fit(T&& X,const CosanMatrix<NumericType>& Y) {
    if (this->MBias==true){
```

Constexpr



```
namespace Cosan{
        template <typename NumericType=std::string,
                  typename = typename std::enable_if<std::is_arithmetic<NumericType>::value,NumericType>::type>
    NumericType StringToNum(const std::string& arg, std::size_t* pos = 0) {
        static_assert(std::is_arithmetic<NumericType>::value, "NumericType must be numeric");
        if constexpr (std::is_same_v<NumericType, unsigned long>) {
            return std::stoul(arg,pos);
        }
        else if constexpr (std::is_same_v<NumericType, unsigned long long>){
            return std::stoull(arg,pos);
        }
        else if constexpr (std::is_same_v<NumericType, int>){
            return std::stoi(arg,pos);
        }
        else if constexpr (std::is_same_v<NumericType, long>){
            .
            .
            .
```

To get on board...

Illustration of a simple machine learning task

- Data Collection
- Data Preparation
- Model Training
- Model Evaluation
- Parameter Tuning
- Make Predictions

Tutorial

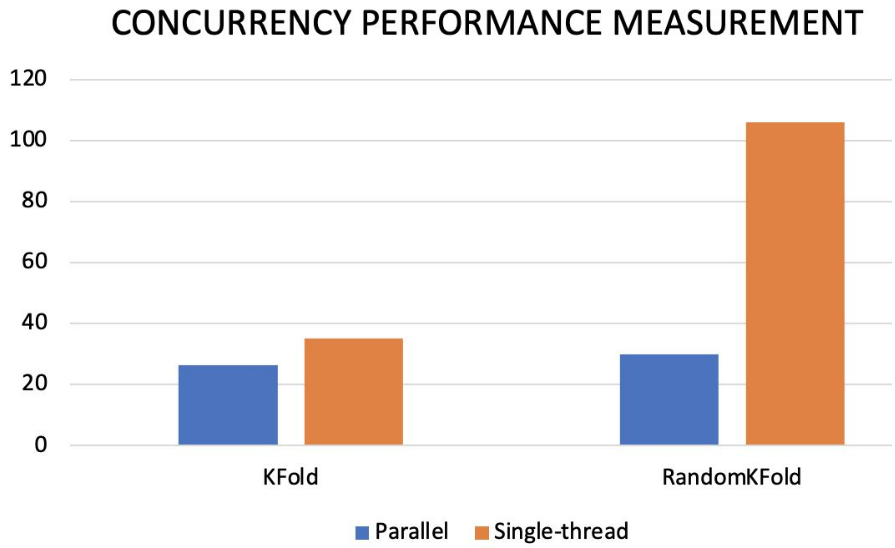
Performance Measurement

100,000 rows

50,000 folds cross-validation

8 cores

measurements in seconds using <chrono>



Future Work & Extensibility

1. Domain knowledge
   - Pipeline
   - Visualization
2. C++ features
   - Import modules
   - Span (input data source)
   - Chrono: timing ()
3. Codebase maintenance
   - Readability and consistency

Q&A