final_project_marker_based

Final Project: PBMC Single-Cell Classification Pipeline¶

Approach 1: Marker-Based Cell Type Annotation example¶

Author¶

Kristof Torkenczy

Computational biologist

Planned and carried out end-to-end

Project Overview¶

Problem statement:¶ In the past 10 years a new frontier of genetics has opened up with single-cell RNA sequencing. Predating these technologies we were able to measure genetic material in bulk, which has resulted in averaged readouts from heterogeneous tissues. With improvements in microfluidic technologies we are now able to measure RNA, and DNA in individual cells within tissues which can reveal answers to complex questions regarding healthy tissues and disease.Of all of the single-cell techniques, single-cell RNA sequencing is the most widely used. In microfluidic devices individual cells are isolated, and RNA molecules are isolated with a matched identifier for the cell. On the computational side RNA molecule sequences (reads) are aligned to the genome (in this case human). Finally, per gene reads are quantified per cell, resulting in a raw counts matrix (gene x cells). These are very 0 inflated (~10% of values in the counts matrix contain non zero values). This necessitates feature selection, followed by PCA, clustering, and finally UMAP to identify structure within the data. One of the active questions currently within this field is how to identify the cell type of each cell. This is usually done via hand annotation by taking gene sets that are "markers" for a cell type. This is a lot of work, so machine learnin, especially transfer learnin has been applied to transfer annotations from reference datasets to novel ones. I decided to try some of what we learnt on this classification problem. The golden standard dataset for single cell RNA

comes from peripheral blood mononuclear cells. These are easy dissassociated cells from the blood, and therefore have gone through extensive analysis.

Objective:¶ Comprehensive machine learning pipeline for automated cell type identification in PBMC single-cell RNA sequencing data using marker-based annotation.

Methodology:¶ The machine learning pipeline employs a marker-based cell type annotation approach, followed by supervised learning for automated cell classification. The methodology consists of four integrated stages:

1. Data Acquisition and Preprocessing

For single-cell data two major software packages exist for analysis: scanpy (in python) and Seurat (in R). Both methods follow the same logic in terms of analysis.

- Download three PBMC datasets from 10x Genomics representing different single-cell technologies
- Extract and standardize data formats (matrix.mtx, features.tsv, barcodes.tsv)

2. Preprocessing and quality control

- **Quality Control**: Calculate QC metrics (number of detected genes/cell, total read counts/cell, percentage of mitochondrial genes).
- **Filtering**: Apply standard filters and show statistics. This is usually dataset dependent. First you want to remove lowly expressed genes (<200 reads aligned), then you remove cells with low number of reads (<500 reads per cell>). Finally you remove cells with high mitochondrial content, which usually indicates that a cell has died (<20% mitochondrial gene).
- **Normalization**: Normalize to sequencing depth followed by multiplying by a fixed number (e.g.: 10k), and log-transform. This is standard procedure as:
  - Each cell in a single-cell experiment can have vastly different sequencing depths (total reads or UMIs), often due to capture efficiency, amplification, or sequencing variation. Without normalization, a cell with twice the sequencing depth would appear to express every gene at twice the level, even if biology is identical.
  - Gene expression counts have a highly skewed distribution: a few genes are expressed at very high levels, while most are low or zero. Downstream statistical methods (PCA, clustering) work better when variance is more symmetric and large values are compressed. Log transformation also helps stabilize variance for high-count genes. Commonly log1p(x) = log(1 + x) is used.
- **Feature Selection**: Find highly variable genes.
  - The variance and mean expression of genes (which is correlated) is modeled and then above than normal genes are selected.

- **Data scaling**: Z-score standardization with maximum value clipping
- **Dimensionality Reduction**: PCA and UMAP visualization

3. Marker-Based Cell Type Annotation We use a simple **clustering-based annotation pipeline** that :

- **Leiden clustering** (resolution=0.5) for initial cell grouping
- **Differential gene expression analysis** to identify cluster-specific markers
- **Automated marker gene enrichment scoring** against known cell type signatures
- **Quality-controlled cell type assignment** with confidence scoring

4. Machine Learning Pipeline

- **Dimensionality reduction**: Apply PCA (50 components) to capture ~20-60% of biological variance based on the union of highly variable genes from all of the studied datasets. Different datasets are projected into the same PCA space. This is done due to the highly 0 inflated and noisy data from single cell datasets.
- **Model ensemble**: Train and evaluate 11 diverse algorithms:
  - **Tree-based**: Random Forest, Gradient Boosting, Decision Tree, AdaBoost
  - **Linear**: Logistic Regression, SVM with RBF kernel
  - **Instance-based**: K-Nearest Neighbors
  - **Probabilistic**: Naive Bayes
  - **Neural Networks**: Multi-layer Perceptron (sklearn), Keras MLP, Keras 1D CNN
- **Transfer learning methods**: k-NN Transfer Learning (custom), Scanpy Ingest (standard)
- **Cross-validation**: 5-fold stratified CV for performance estimation
- **Comprehensive metrics**: Accuracy, Precision, Recall, F1-Score, Specificity, ROC-AUC, PR-AUC
- **Feature engineering**: Union HVG strategy for optimal cross-dataset compatibility

5. Validation Strategy

- **Same-dataset validation**: Train/test split (70/30) on single dataset for method optimization
- **Cross-dataset validation**: Train on one dataset, test on independent datasets for generalization assessment
- **Performance metrics**: Accuracy, precision, recall, ROC-AUC,PR-AUC, F1-score, and confusion matrices

Datasets:¶ My analysis uses three high-quality PBMC datasets from 10x Genomics, representing different single-cell technologies:

PBMC 3k Dataset (Training)

- **Source**: 10x Genomics public dataset
- **Technology**: Chromium Single Cell 3' v1
- **URL**: `https://cf.10xgenomics.com`
- **Format**: Uncompressed 10x format (older version)
- **Files**: `matrix.mtx`, `genes.tsv`, `barcodes.tsv`
- **Extraction**: Direct download and extraction to `data/pbmc3k_extracted/`
- **Size**: ~2,700 cells × 32,738 genes
- **Notes**: Well-characterized reference dataset, commonly used in tutorials

PBMC Multiome Dataset (Test)

- **Source**: 10x Genomics public dataset
- **Technology**: Chromium Single Cell Multiome ATAC + Gene Expression
- **URL**: `https://cf.10xgenomics.com`
- **Format**: Compressed 10x format (newer version)
- **Files**: `matrix.mtx.gz`, `features.tsv.gz`, `barcodes.tsv.gz`
- **Extraction**: Download, decompress, filter to RNA-only features
- **Feature Types**: 36,601 Gene Expression + 143,887 ATAC Peaks (RNA-only used)
- **Size**: ~11,898 cells × 36,601 genes (RNA features only)
- **Notes**: Granulocyte-sorted PBMCs, multimodal dataset with ATAC+RNA

PBMC 10k v3 Dataset (Test)

- **Source**: 10x Genomics public dataset
- **Technology**: Chromium Single Cell 3' v3 Chemistry
- **URL**: `https://cf.10xgenomics.com`
- **Format**: Compressed 10x format
- **Files**: `matrix.mtx.gz`, `features.tsv.gz`, `barcodes.tsv.gz`
- **Extraction**: Download and decompress (RNA-only dataset)
- **Feature Types**: 33,538 Gene Expression features only
- **Size**: ~11,769 cells × 33,538 genes
- **Notes**: Standard 3' v3 chemistry dataset

System Architecture¶ The pipeline implements an architecture with three core components working in sequence to deliver robust single-cell classification results.

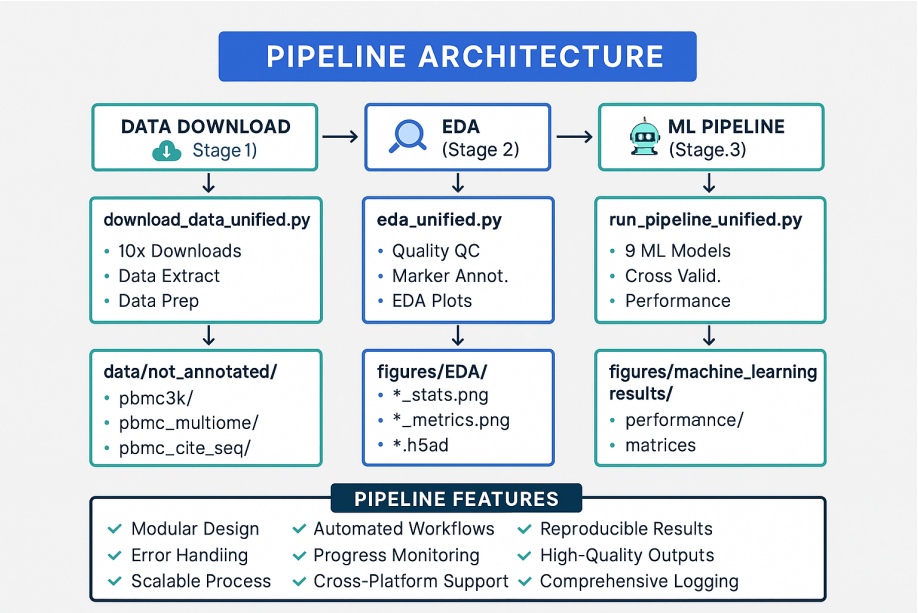

Figure 1. Block design of pipeline

Component Descriptions:

Stage 1: Data Download (`download_data_unified.py`)

- **Purpose**: Automated acquisition and standardization of 10x Genomics datasets
- **Functions**:
  - Downloads compressed archives from 10x servers with progress tracking
  - Extracts and validates file integrity (matrix.mtx, features.tsv, barcodes.tsv)
  - Filters multimodal data to RNA-only features
  - Organizes data into standardized directory structure
- **Output**: Raw datasets in `data/not_annotated/` directory
- **Data format**
  - **matrix.mtx**:

This file contains the sparse count matrix, representing the gene expression levels. Each row corresponds to a feature (like a gene). Each column corresponds to a cell barcode. The numerical values in the matrix are the UMI counts. - **features.tsv** (or features.csv): This tab-separated file provides information about the rows (features) of the matrix.mtx file. The first column is the feature identifier (e.g., gene ID). The second column is the gene symbol (a common name). Subsequent columns might contain other metadata, depending on the specific data type. - **barcodes.tsv** (or barcodes.csv): This tab-separated file provides information about the columns (cells) of the matrix.mtx file. Each

entry in this file is a cell barcode. **Note**: This approach serves as a baseline for comparison with Approach 2, which uses pre-annotated reference datasets from SeuratData.

Stage 2: EDA and Annotation (`eda_unified.py`)

- **Purpose**: Exploratory analysis and marker-based cell type annotation
- **Functions**:
  - Quality control filtering (genes/cells thresholds)
  - Marker-based annotation using immune cell signatures
  - Comprehensive visualization generation (statistics, quality metrics)
  - Data preprocessing for machine learning (normalization, scaling)
- **Output**: Annotated datasets (`.h5ad` format) and EDA visualizations

Stage 3: Machine Learning (`run_pipeline_unified.py`)

- **Purpose**: Comprehensive ML analysis with multiple algorithms and validation modes
- **Functions**:
  - Trains 12 diverse ML algorithms with hyperparameter optimization
  - Performs same-dataset and cross-dataset validation
  - Generates performance metrics and publication-quality visualizations
  - Statistical analysis with confidence intervals
- **Output**: Performance reports, confusion matrices, model comparisons

1. Data Download and Organization¶

Download and organize 10x Genomics PBMC datasets for marker-based annotation analysis.

In [27]:

```
# Download not_annotated (10x) data
# -q is for quiet mode -v is for verbose mode
!python download_data_unified.py --approach not_annotated -q
```

2. Exploratory Data Analysis and Marker-Based Annotation¶

Perform comprehensive EDA and apply marker-based cell type annotation to the 10x datasets.

In [28]:

```
# Run EDA with marker-based annotation
!python eda_unified.py --approach not_annotated -q
```

```
WARNING: It seems you use rank_genes_groups on the raw count data. Please logarithmize your
WARNING: It seems you use rank_genes_groups on the raw count data. Please logarithmize your
```

Exploratory Analysis Results¶

The EDA shows that all datasets have at least 7 cell types based on marker based analysis. Also, the median reads per cell (total counts per data point) is above 500 for all data sets, which is good. Similarly, the number of genes (features) with at least 200 reads is good. Similarly, removed cells that had a high (>20%) mitochondrial fraction due to them being likely dead cells.

Dataset Overview¶

Dataset	Cells	HVGs*	Cell Types	Median UMI/Cell**	Median Genes/Cell**	Median
PBMC 3k	2,698	2,000	7	2,198	817	2.0%
PBMC Multiome	11,621	2,000	10	3,812	1,840	9.7%
PBMC CITE-seq	7,798	2,000	9	3,661	1,397	5.9%

Legend:

- **HVGs*** = Highly Variable Genes selected for analysis (from ~32,000 original genes)
- **All cell-level metrics**** (UMI/cell, Genes/cell, MT%) = Calculated on original data before HVG selection

Note: Quality control metrics are always computed on the full original dataset to accurately assess sequencing quality and cell viability.

PBMC 3k Annotation Results¶

This is the example QC for the reference data. We have the same results for the two test data as well.

Quality Control Assessment¶ The basic stats of the training sample.

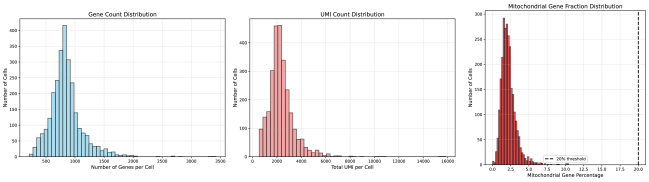

Figure 2. Basic statistics of PBMC3k (training data)

Data Quality Metrics summary:

- **Mitochondrial contamination**: Low levels (below <20%)
- **Gene detection**: Relatively lower number of genes with reads (median 817 genes/cell)
- **Library complexity**: Appropriate UMI counts (median 2,198 UMIs/cell)

- **Normalization**: all necessary normalization steps done for downstream analysis

Cell-type annotation¶ Our marker-based approach successfully identified **7 distinct cell populations**:

Cell Type	Count	Percentage	Key Markers	Confidence
CD4+ T cells	1,212	44.9%	RPS12, LDHB, RPS25	High
CD14+ Monocytes	651	24.1%	FTL, FTH1, TYROBP, LYZ	High
NK cells	430	15.9%	NKG7, CST7, GZMA, CTSW	High
Naive B cells	348	12.9%	CD74, CD79A, HLA-DRA	High
Dendritic cells	36	1.3%	HLA-DPA1, HLA-DPB1	Very High
Platelets	13	0.5%	GNG11, SDPR, PF4, PPBP	Very High
Unknown	8	0.3%	TYMS, KIAA0101, ZWINT	Low

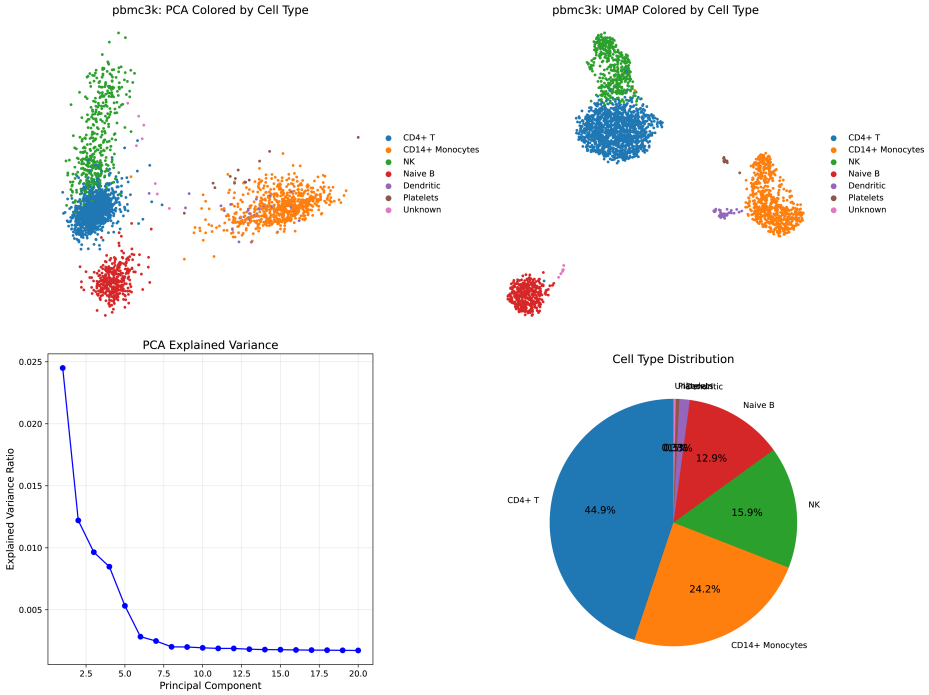

Figure 3. Cell types and dimensionality reduction in PBMC3K

Cell annotation and dimensionality reduction summary:

- **PCA and UMAP**: Shows cells based on first two PCAs, then in UMAP space. Clearly cell types separate well.
- **PCA variance plot**: Shows how the first 8 PCA captures most of the variation in the data.

- **Cell types**: Cell type annotation proportions follow known biology of this dataset.

Cross-Dataset Comparison¶

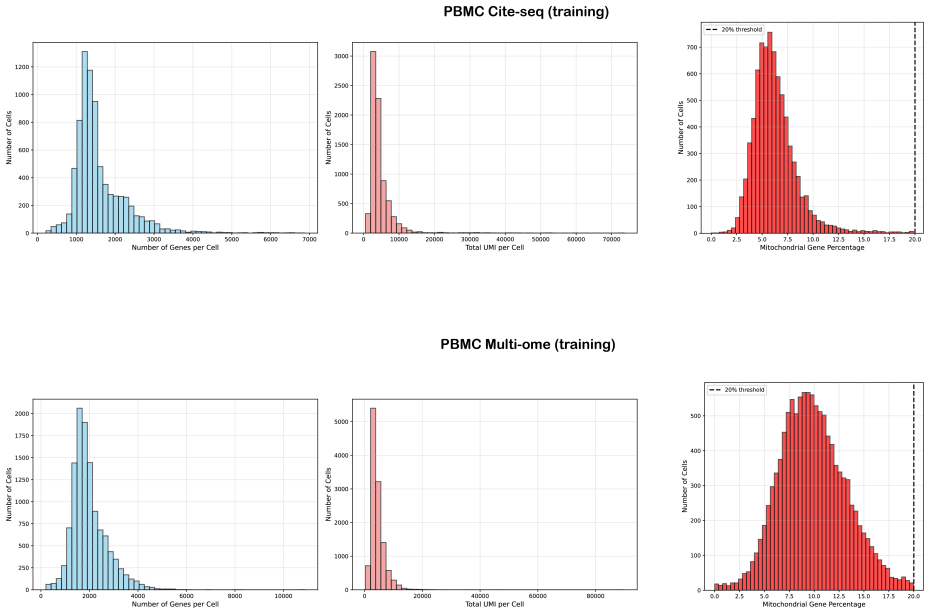

Figure 4. Testing dataset QC statistics

- **Mitochondrial contamination**: Higher level across all datasets, but still acceptable (5.9%-9.7%)
- **Gene detection**: Higher gene capture than training (>1000 median genes/cell)
- **Library complexity**: Higher UMI counts than training (~3000 median UMIs/cell)

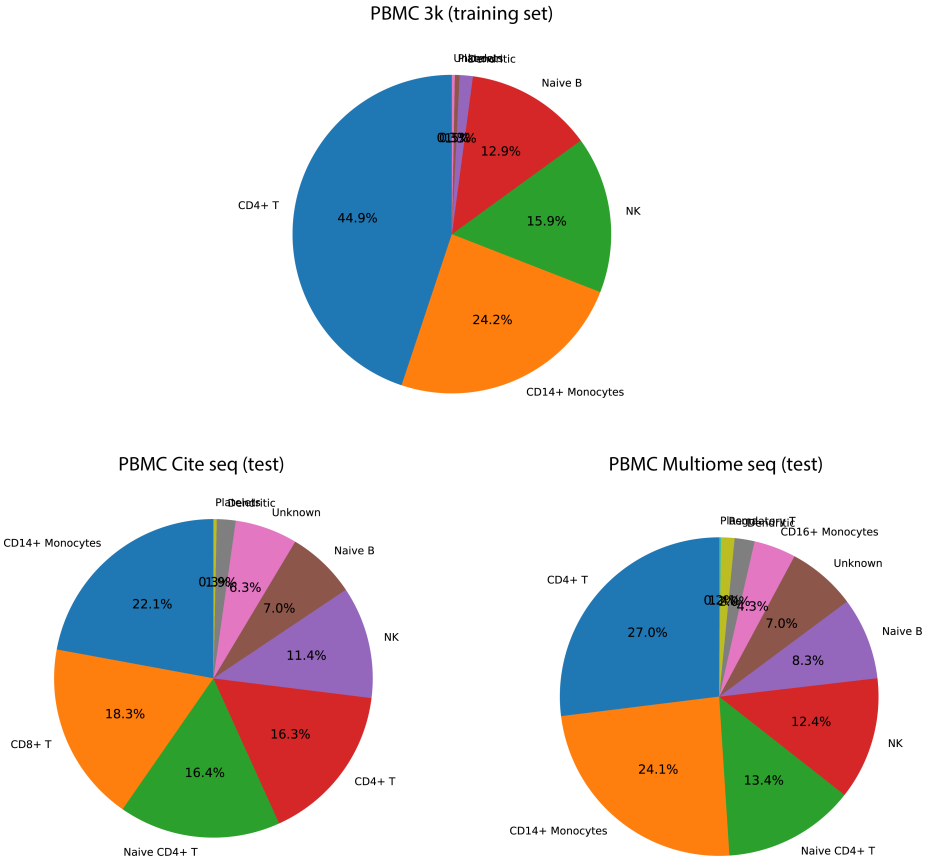

Figure 5. Cell types across all three datasets

- PBMC Multiome shows highest diversity (10 cell types) due to larger sample size
- CITE-seq reveals additional T cell subtypes (CD8+ T, Naive CD4+ T)
- Core immune populations consistent across all platforms

Platform Differences:

- **10x 3' v3** (PBMC 3k): Standard scRNA-seq, good for basic annotation
- **10x Multiome**: Enhanced gene detection, captures rare populations
- **CITE-seq**: Surface protein information enables finer T cell resolution

The comprehensive EDA establishes a robust foundation for downstream machine learning analysis, with high-quality, well-annotated datasets ready for predictive modeling.

3. Machine Learning Pipeline¶

Train and evaluate 11 machine learning algorithms on marker-annotated data with both same-dataset and cross-dataset validation.

In [ ]:

```
# Run complete ML pipeline
!python run_pipeline_unified.py --approach not_annotated --mode both -q
```

4. Machine Learning Results¶

Our machine learning pipeline evaluated 13 different algorithms on the marker-annotated PBMC datasets, including traditional ML methods (9), deep learning models (2), and transfer learning approaches (2). In the end we compared to the ingest method in scanpy, which is widely used in the single-cell field.

Same-Dataset Performance¶

Top Performing Models (PBMC 3k - Internal Validation):

Model	CV Accuracy	Test Accuracy	Precision	F1-Score	ROC-AUC
Logistic Regression	**97.6%**	**97.9%**	97.9%	97.8%	99.8%
Keras MLP	98.0%	97.6%	97.7%	97.6%	99.8%
SVM (RBF)	97.8%	97.6%	97.7%	97.5%	99.9%
Neural Network	98.1%	97.5%	97.6%	97.5%	99.9%
Keras 1D CNN	97.9%	97.0%	97.0%	97.0%	99.9%

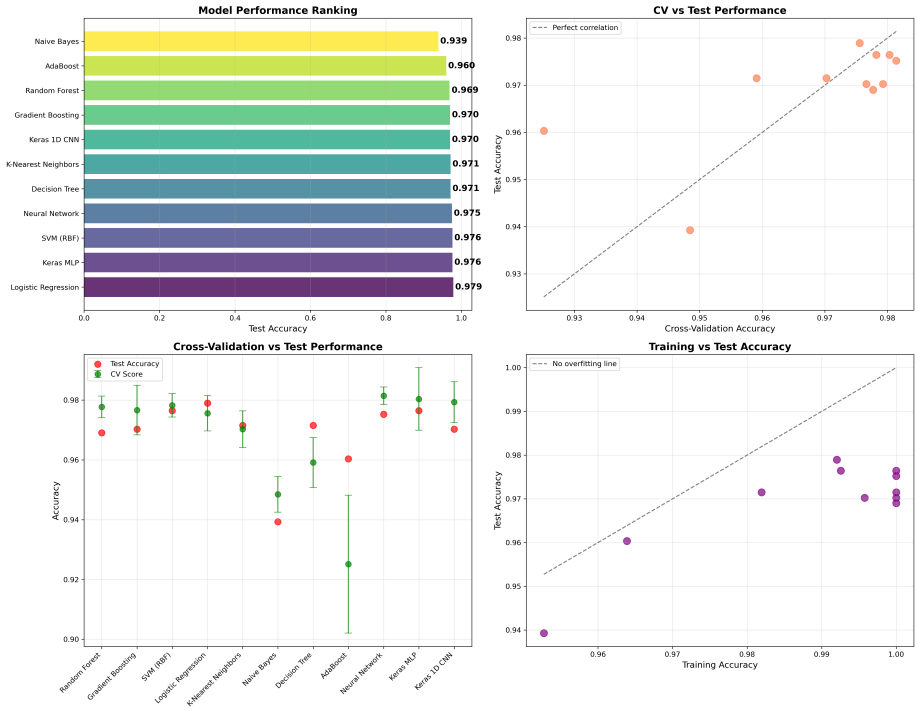

Figure 6. Within datasets training, testing and cross validation

Key Observations:

From figure 6 we can see several things.

- **Logistic Regression leads**: Achieves highest test accuracy (97.9%) with good test generalization. This still might be overfitting.
- **Consistent performance**: Deep learning models show strong but slightly lower test accuracies. I wonder how they will generalize to other datasets.
- **Less overfitting**: CV scores align well with test performance across all models.
- **Robust classification**: All top models achieve >97% accuracy with very good ROC-AUC scores.

Because of the class imbalance of the annotated cells we can also look at the F1 scores and specificity across the cell types. Interestingly, it shows that the models are relatively good at detecting true negatives, while balancing precision (few false positives) and recall (few false negatives). This is also not equally spread across cell types. For instance Dendritic cells seem to be harder to predict. This makes sense as they are very rare in the dataset.

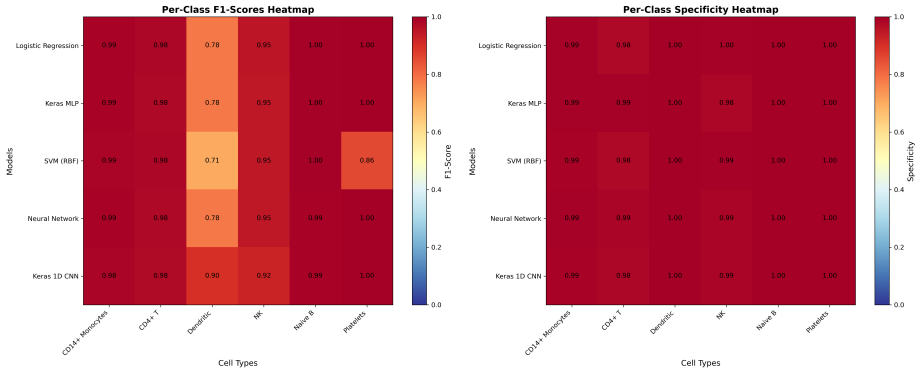

Figure 7. F1 scores per cell type label

Cross-Dataset Performance¶

External Dataset Testing (Models trained on PBMC 3k → Tested on external datasets):

I looked at precision, recall, and F1 scores first. Then extended the method to ROC_AUC, and PR_AUC. The top 5 models across all 11 models is shown here.

PBMC CITE-seq Results:¶

Model	Test Accuracy	Precision	F1-Score	ROC-AUC
Naive Bayes	**98.9%**	98.9%	98.9%	99.9%
SVM (RBF)	98.7%	98.7%	98.5%	99.9%
Keras 1D CNN	98.5%	98.5%	98.5%	99.9%
K-Nearest Neighbors	98.6%	98.6%	98.5%	99.5%
Neural Network	98.3%	98.4%	98.2%	99.9%

On first glace we can see is that the models generalize really well to the PBMC CITE-seq data. Lets dig into the stats a little.

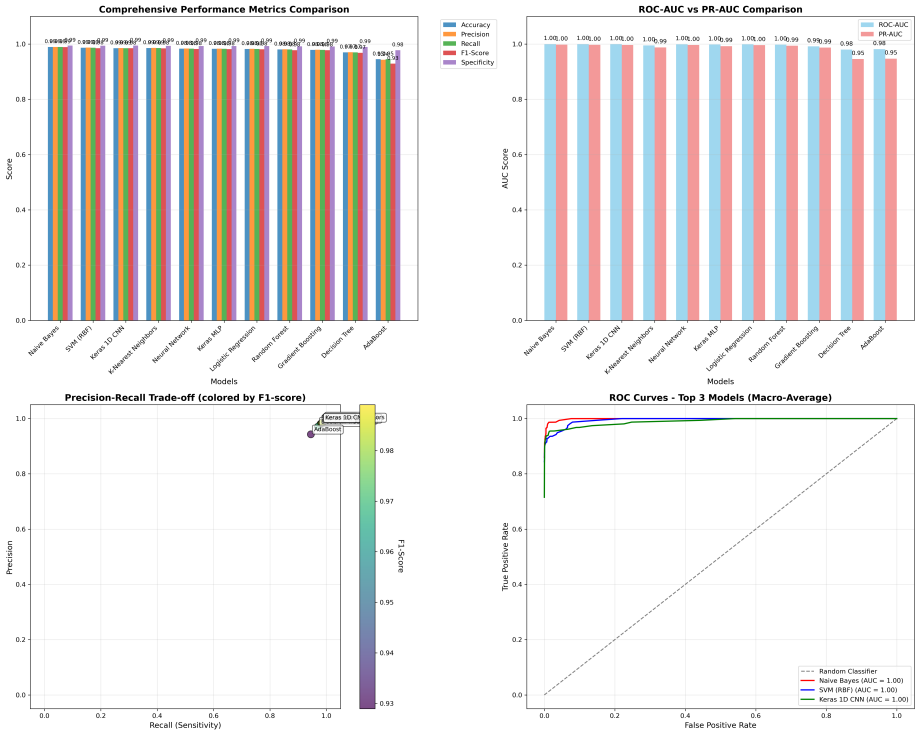

Figure 8. Evaluation metrics for classification on the PBMC CITE-seq dataset

By the looks of it Naive bayes does really well. Logistic regression less so which could indicate that that logistic regression was overfit on the training data.

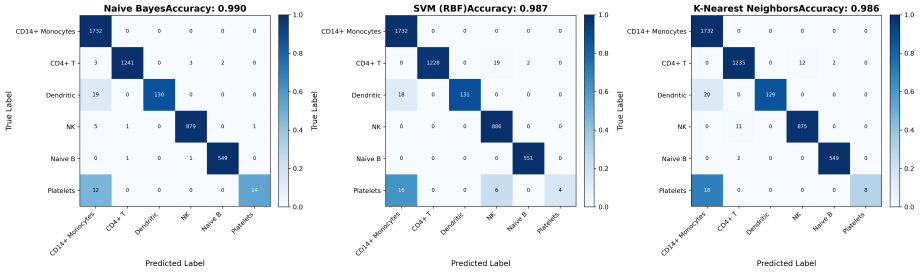

Figure 9. Confusion matrix for classification on the PBMC CITE-seq dataset

Clearly platelets are a difficult rare class to predict. Now lets look at the Multi omic results.

PBMC Multiome Results:¶

Model	Test Accuracy	Precision	F1-Score	ROC-AUC
Keras 1D CNN	**97.5%**	97.6%	97.5%	99.8%
Logistic Regression	97.5%	97.5%	97.4%	99.9%
Neural Network	97.3%	97.3%	97.2%	99.8%
Keras MLP	97.3%	97.3%	97.2%	99.7%
SVM (RBF)	97.3%	97.3%	97.2%	99.9%

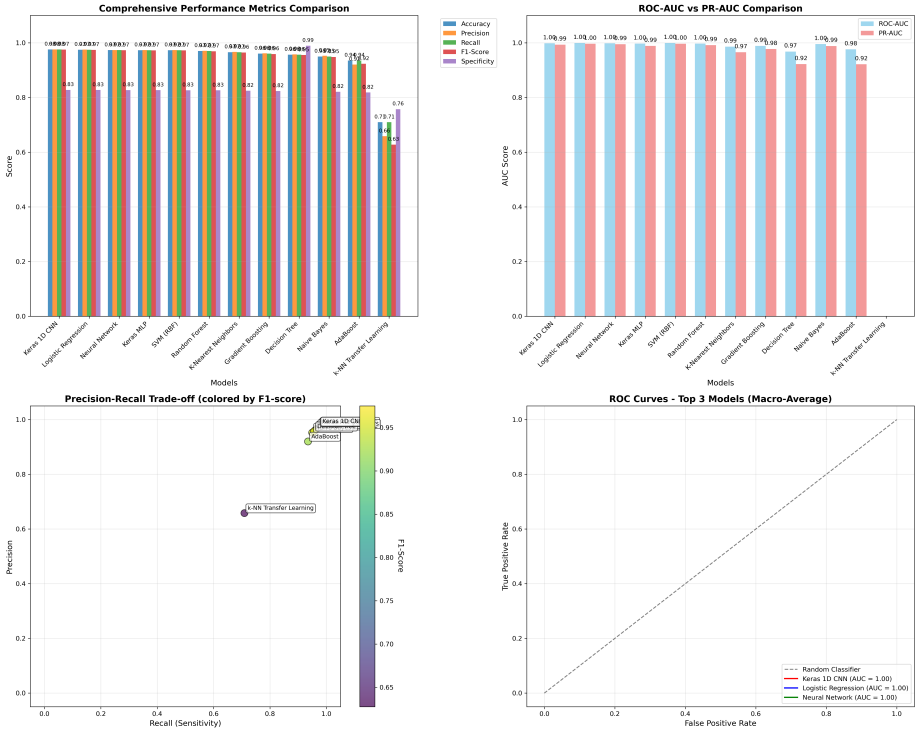

Figure 10. Evaluation metrics for classification on the PBMC multi-ome dataset

One of the interesting results from this is that in this we have:

- **High F1** You have a good balance of precision and recall for the positive class. The model correctly identifies many true positives and doesn't make too many false positives relative to the number of positives.
- **Lower specificity** The model is not as good at identifying negatives. It's producing more false positives, which hurts its ability to correctly label negatives as negatives.

Which just shows that models do not generalize universally well to all datasets.

Transfer Learning Performance¶

Specialized Transfer Learning Methods:

Method	CITE-seq Accuracy	Multiome Accuracy	A
k-NN Transfer Learning (Custom Implementation)	53.3%	65.9%	R
Scanpy Ingest Transfer Learning (Standard Method)	54.7%	88.5%	S

Key Findings:

- **Scanpy Ingest excels on Multiome**: 88.5% accuracy shows strong manifold-based transfer.
- **Both methods struggle with CITE-seq**: ~54% accuracy indicates biological differences.
- **Traditional ML performs better**: Union HVG strategy (97-99%) significantly outperforms transfer learning
- **indication** We might be overfitting here or our marker based annotation is not actually biologically precise.

Technical notes¶

The bettter performance of traditional ML over transfer learning could be because:

1. **Union HVG Strategy**: ML methods use 2,947 common highly variable genes across datasets for optimal cross-dataset compatibility
2. **Feature Quality > Quantity**: Curated Union HVGs (2,947) full overlap (~11,700 genes)
3. **Transfer Learning Limitations**: Even with maximum gene overlap, transfer learning achieves 54-88% vs traditional ML's 97-99%
4. **Biological Insight**: Feature engineering and preprocessing can be more impactful, but can overfit

Summary and next steps:¶

Cross-Dataset Analysis:

- **High accuracy**: 97-99% accuracy across different platforms with Union HVG strategy
- **Robust across platforms**: Consistent high performance on CITE-seq (98.9%) and Multiome (97.5%)

Transfer Learning:

- **Traditional ML performs better**: Union HVG strategy outperforms specialized transfer learning methods
- **Feature selection**: Feature selection with clean data beats sophisticated transfer methods

- **Methodological insight**: k-NN transfer (70.9% on Multiome) demonstrates value of proper preprocessing

I am worried that I may have done some major overfitting here in terms of biology, if not technically. I would keep testing to see what has happened but I think I have to stop here. I have added a funtionality to the pipeline that downloads pre-annotated data. See expert annotations ipynb.

Repository location:

https://github.com/torkencz/Stanford_TECH_27_final_project_Torkenczy