

CMSE 381 : Final Project

INTRODUCTION

The goal of the project is to take the RNA as input and predict the protein. Gex is a gene expression which can also be RNA and ADT is the protein.

The GEX data is pre-processed and contains gene expression which was measured using 3' capture of single-cell RNA. Cells were filtered based on mitochondrial content, UMI counts per cell, and genes detected per cell.

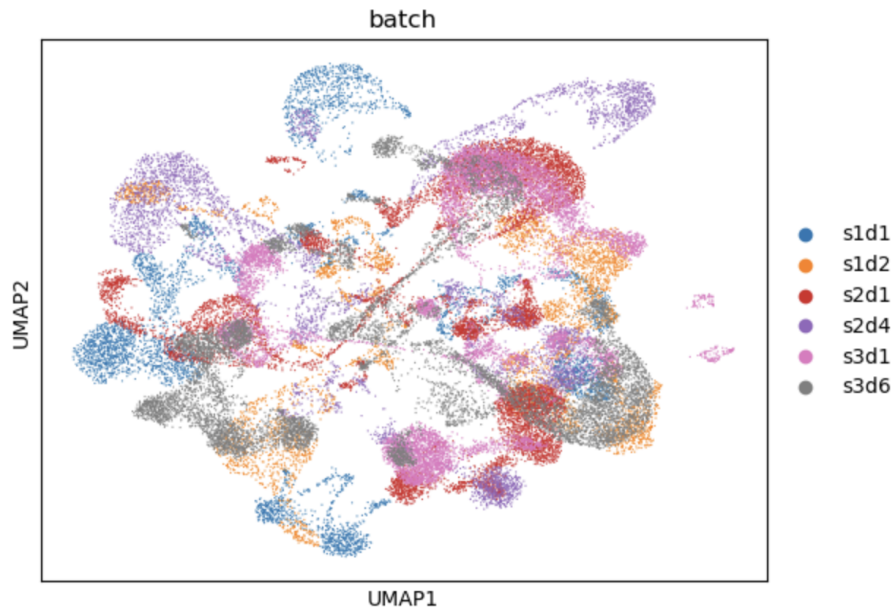


Figure 1. The batch in GEX

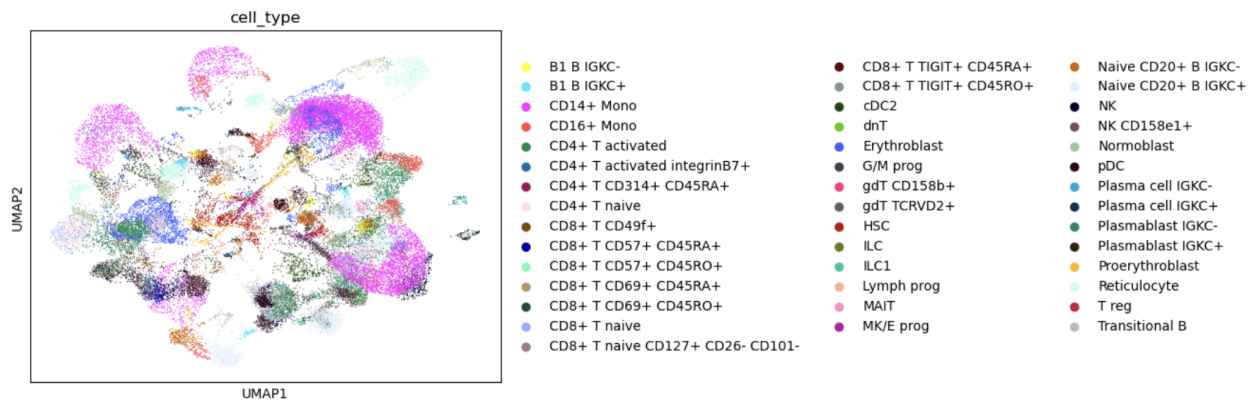


Figure 2. The cell type in GEX

The protein data contains 134 cell surface markers and 6 isotype controls.

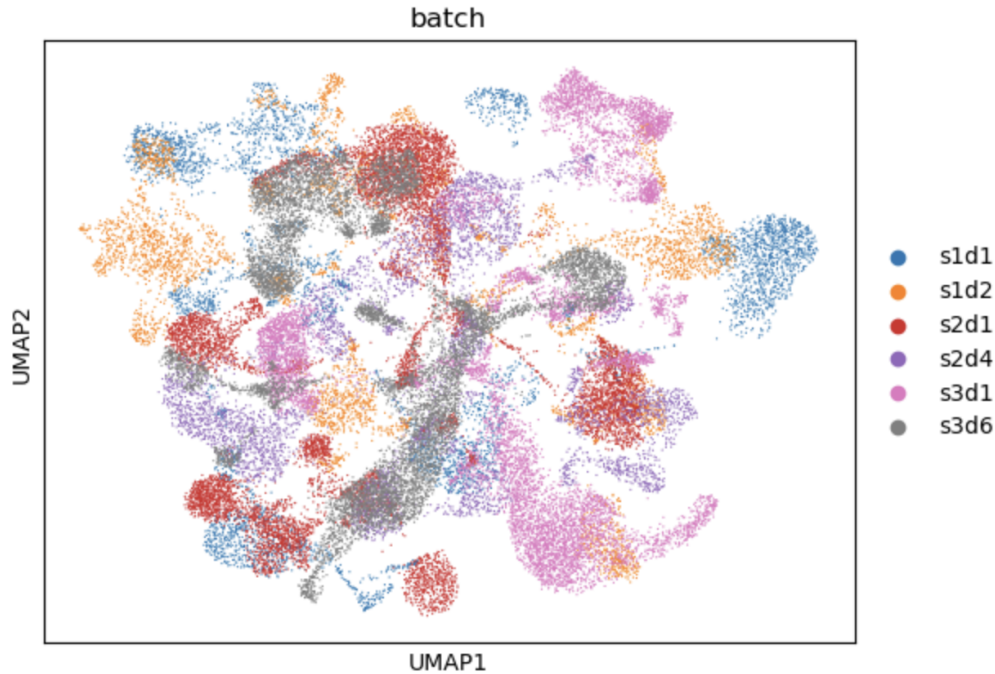


Figure 3. The batch in ADT

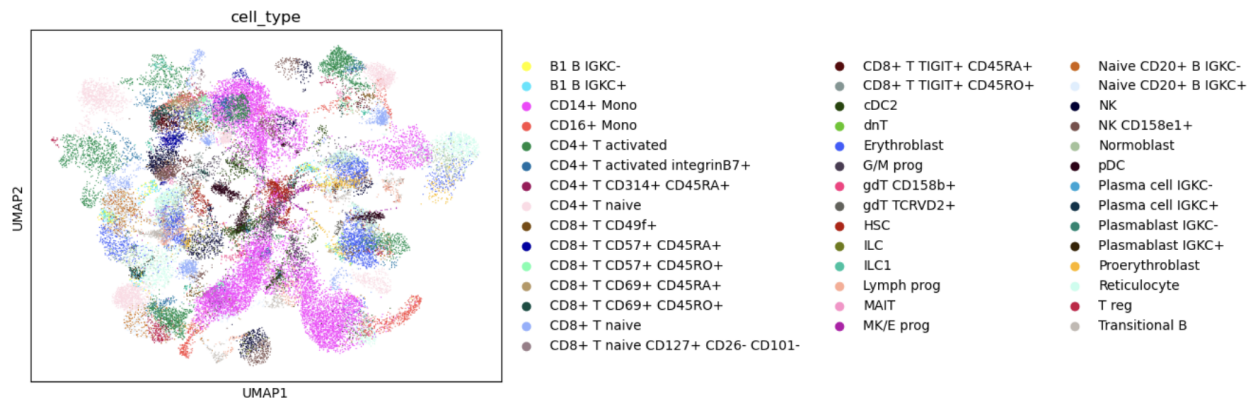


Figure 4. The cell type in ADT

The ultimate goal is to create a prediction model that takes RNA as input and predicts the protein abundance.

METHODS

For this project, the gex train and test data, as well as the adt test and train data is stored in Anndata files. Anndata files are a file format used in single-cell genomics research to store and manipulate large-scale gene expression data. The format is designed to represent gene expression data along with additional metadata, such as cell type or experimental condition. These files are stored in a hierarchical data structure called a "AnnData object".

The dataset may contain a large number of missing values. In order to get an accurate model, we must impute the data to fill in missing values using the remaining values. I chose a KNN Imputer in order to input all 4 files.

KNN Imputer

The KNN Imputer uses the values of the k-nearest neighbors to estimate the missing values in a dataset. KNN Imputer was specifically chosen as its algorithm can work with both categorical and numerical data and does not require assumptions about the distribution of data. Missing values can be filled in a way that takes into account the relationships between different variables in the dataset. The KNN Imputer uses the values of the k-nearest neighbors to estimate the missing values in a dataset.

```
# Impute missing values using KNN imputation
imputer = KNNImputer(n_neighbors=5)
train_gex.X = imputer.fit_transform(train_gex.X.toarray())
test_gex.X = imputer.transform(test_gex.X.toarray())
```

Figure 5. Imputing the data.

The KNN imputer is very computationally expensive and can take a while to impute all the data. To save time, we use the python module 'Pickle' to save the data in a serialized format. This is saved to the directory in .pkl files. Note that we do not impute the ADT data as that is our result/y variable.

PCA

After fitting the PCA, the code calculates the cumulative explained variance ratio for each principal component using the `explained_variance_ratio_` attribute of the PCA object. This plot can be used to determine the number of principal components to retain for further analysis or to identify patterns in the data that are captured by the principal components.

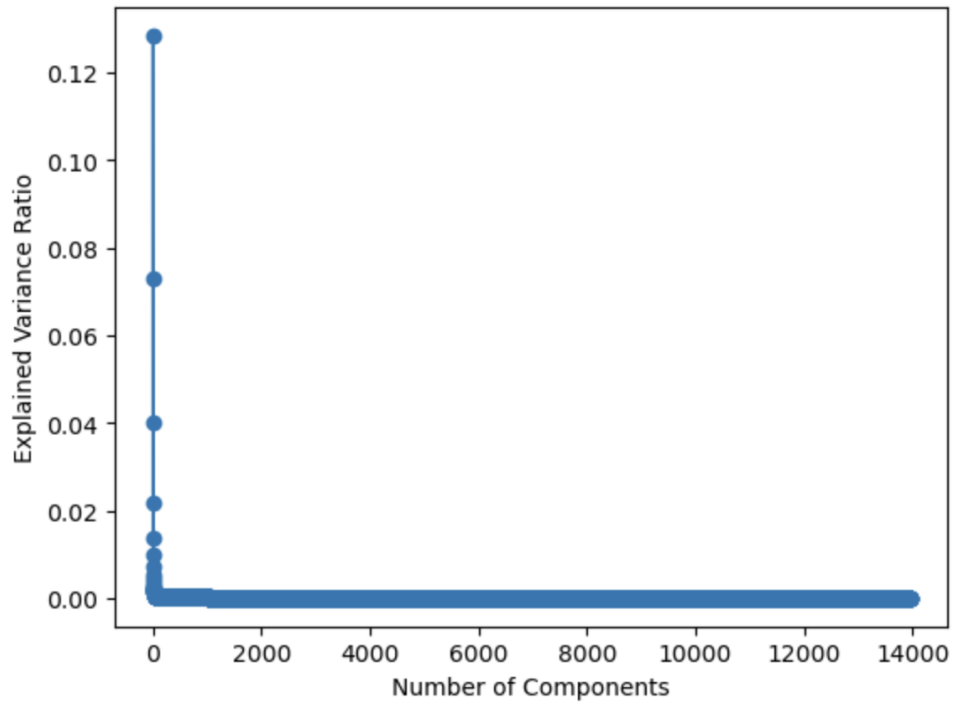


Fig 6. Explained variance ratio vs Number of components

Here, unfortunately we cannot determine the number of components. Lets zoom in.

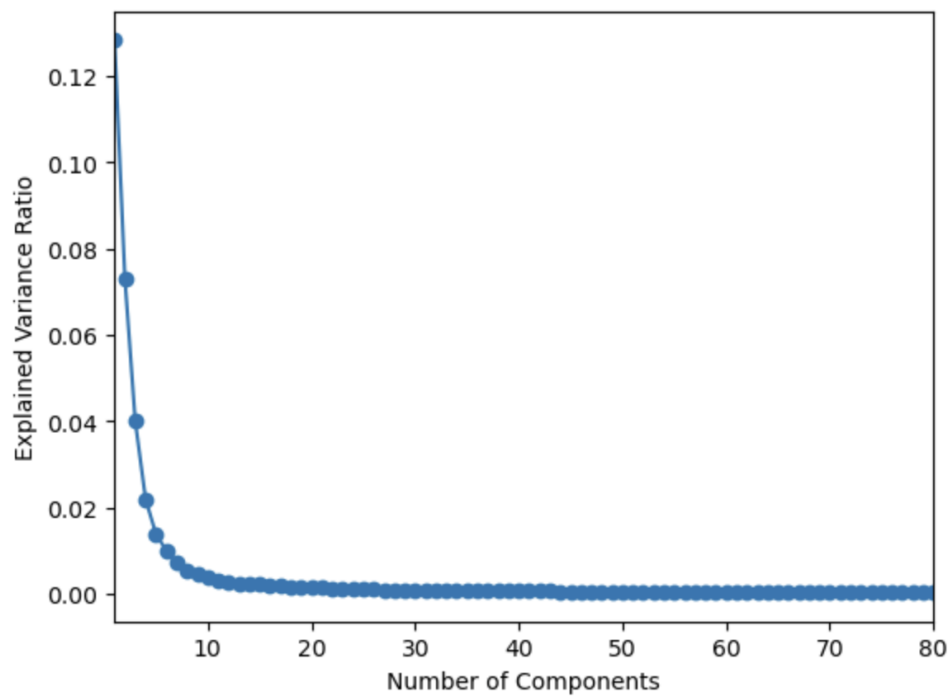


Fig 7. Explained variance ratio vs Number of components zoomed in

Based on the graph, we use 50 as the number of components in further PCA analysis.

MLP Regression

MLP regression refers to a type of regression analysis that uses a multi-layer perceptron (MLP) neural network to predict a continuous target variable based on a set of input features. Two hidden layers with 50 neurons each were used for this model.

RESULTS

Let us look at a graph of the actual data vs the predicted data.

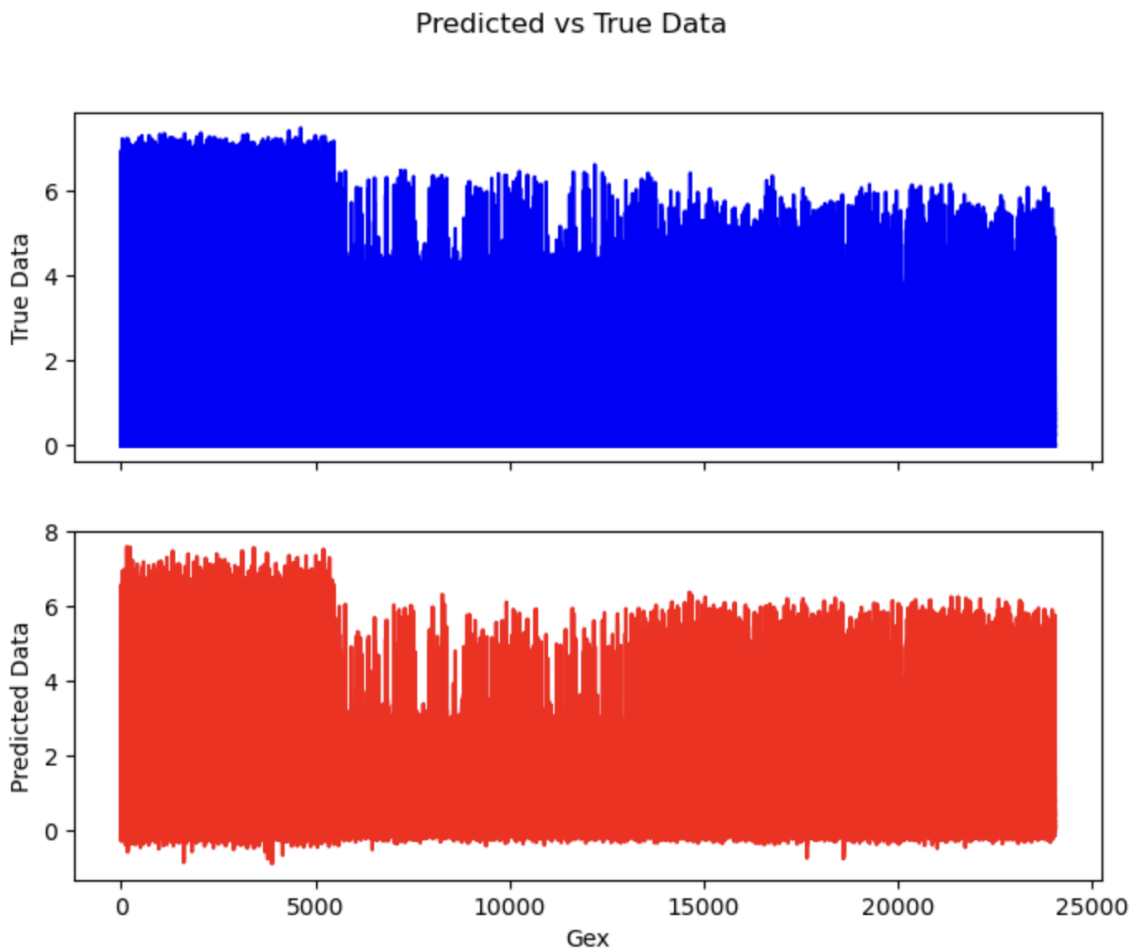


Fig 8 : actual and predicted data

The RMSE is also a very important aspect in determining the accuracy of a model. It measures the difference between the actual and predicted data. It shows how far the prediction is from the actual values using Euclidian distance. A lower RMSE indicates that the model is better at predicting the target variable.

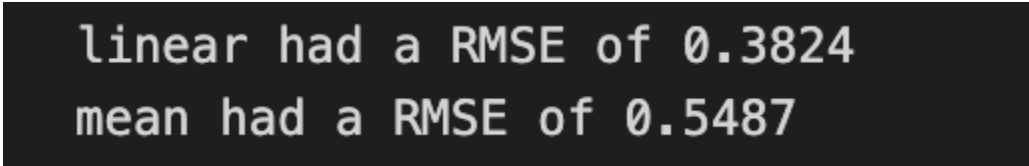
```
from sklearn.metrics import mean_squared_error
mean_squared_error(true_test_adt.X.toarray(), pred_test_adt.X, squared=False)
0.35555059444143955
```

Fig 9 : RMSE of Ishita Kokil's model

You can see that the bagged model had a very slightly better performance.

The RMSE of the model in this project is 0.35.

Let us compare this RMSE to the baseline rmse and the linear rmse that were previously provided:



linear had a RMSE of 0.3824
mean had a RMSE of 0.5487

Fig 10: RMSE of linear and baseline model.

As it can be seen, my model outperforms both the linear and the baseline model, making it a more accurate way of predicting the protein based on the gene data.