Final feature selection/processing:

The dataset consisted of 101461 features. On analyzing the dataset we found that a large number of rows consisted of NaN Values which means the data field is undefined which makes the data very sparse. Handling NaN Values and missing data is the most crucial part of data processing.

When we train and test data using Machine Learning models we do not want any NaN Values to predict accurately. So, the first step for feature processing was dropping all the rows that consisted of NaN Values as they do not contribute to the analysis. Hence, we found that there were a total 90,826 observations which had at least one NaN value.

Thus, we reduced the features to 10,635 by completely removing the NaN values and missing data. Now, the dataset is cleaned to perform processing. The shape of the input vector is now reduced to (90, 10,635).

• T-test

We split the data into the 75% training data and 25% testing data after the elimination of the Nan rows i.e, on 10,635 features. Once we have our training data, we labeled it into mainly four categories as follows: Healthy, Non-severe-COVID-19, Severe-COVID-19, and Symptomatic-non-COVID-19 where they are labeled as Label 1, Label 2, Label 3 and Label 4 respectively. Then, we applied the t-test on the training data, that is, for each feature, we first grouped the rows by labels and then formed combination pairs of these labels in the following manner: $(Label\,x,\ Label\,y)$ where $x,\ y \in \{1,\ 2,\ 3,\ 4\}$. For each such combination of labels, we applied t-test using scipy.stats library.

The t-test measures the difference between the two samples by arithmetic means. The p-value expresses the likelihood of seeing one or more extreme values if the null hypothesis, that is in our case the labels from each combination have the same means, is true. A p-value greater than a set threshold (in our case, 5%) implies that our finding is less likely to have happened by chance. As a result, the null hypothesis of equal means for the said combination of labels is not rejected, which means that all the labels have similar mean. Therefore, we entirely remove that particular feature from our dataset. However, if the p-value is less than our threshold then we have evidence against our null hypothesis of equal means, which implies that one of the combinations of labels have a different mean so we kept that feature in our dataset. After performing t-test on our training data we reduced down to 1595 features. So using the t-test function we transform the data with shape (66, 10635) to (66, 1595).

• Correlation

The next step after reducing the number of features using t-test is to find the correlation of each peptide with every other peptide in order to get rid of highly correlated features. We computed the correlation matrix for the reduced dataset of 1595 features.

Initially, in order to get a sense of the correlation between the features, we implemented a simple function that would, for each row of the correlation matrix, filter out those peptides with the maximum positive and negative correlation. We used a counter to increment the count of these peptides. In the end, we were left with a list of very few peptides(24) with the highest count. We now knew which features were important and had to be included as inputs but this many peptides would not be sufficient enough to accurately classify the labels.

However, in order to obtain the entire subset of features that had to be given as inputs to our model, we used .corr() to find the pairwise correlation of all columns and then remove all the features that have values greater than 0.9 which we set as our threshold value. Our input before applying correlation had the shape (66, 1595) and then the output shape is (66, 945) which means the input vector to the model would have 945 features.

Model details

- ### Logistic Regression

Logistic Regression is used in statistical software to understand the relationship between the dependent variable and one or more independent variables by estimating probabilities using a logistic regression equation. This type of analysis can help you predict the likelihood of an event happening or a choice being made. It can be used for Classification as well as for Regression problems, but mainly used for Classification problems. Since our dataset contains numerical values, we use Logistic regression as one of the models because logistic regression works well with numerical data.

We apply this model once we have performed data processing and cleaning and also completed feature extraction and reduced our number of features to the 945 most important features.

- ### Random Forest

The presence of numerous features maximizes the performance of random forest algorithms and hence, minimizes the generalization errors, while providing apt split selection between them. The random forest technique can also handle big data with numerous variables running into thousands.

As we have a high number of input features, it can automatically balance data sets when a class is more infrequent than other classes in the data. This was the primary reason for implementing Random Forest Supervised Learning Algorithm on the labeled train data and unseen test data. These classify the tuples, in multiple classes based on various parameters like Gini Index for Random Forest. We apply this model once we have performed data processing and cleaning and also completed feature extraction and reduced our number of features to the 945 most important features.

In order to optimize the random forest algorithm, we plan to use hyperparameter tuning. We perform many iterations of k-fold cross-validation, each time using different model settings. We then compare all of the models, select the best one, train it on the full training set, and then evaluate on the testing set.

We will select the model that gives us the highest accuracy along with the highest recall and precision score. As recall and accuracy are the main output parameters we are concerned with, a tradeoff between these two will decide the best performance model to be selected.

Model Optimization using RFECV

RFE is a feature selection method that works like a wrapper. This indicates that a different machine learning algorithm is given and employed in the method's core, which is wrapped by RFE and used to aid with feature selection. RFE works by starting with all features in the training dataset and successfully deleting features until the target number remains. This is accomplished by re-fitting the model using the given machine learning method, ranking features by significance, removing the least important features, and re-fitting the model. This method is continued until a specified number of features remains.

The specified machine learning model (that is using logistic regression as the estimator) is used to score features. In our project, we achieve this using the sklearn.feature_selection library. To automatically select the number of features chosen by RFE, we use the RFECV class. This can be achieved by performing cross-validation evaluation of different numbers of features and automatically selecting the number of features that resulted in the best mean score.

So the number of input features is 945 and as mentioned the RFECV itself chooses the optimal number of features using cross-validation. For Logistic regression, there are 809 features. For the Random feature classifier, there are 934 features and for the Support vector machine classifier, the number of features are 800 features. These values are considering the 75%-25% split.

Evaluation of the model(s) by their classification results

We split the data into the 75% training data and 25% testing data after the elimination of the Nan rows i.e, on 10,635 features, after performing T-test i.e., 1595 features and after correlation i.e. 945 features. The parameters used for the RFECV model are as follows:

- **estimator**: This parameter is used to specify the machine learning model we are using. We pass the machine learning that we need to wrap into the RFECV model. We used three estimators here: SVM, Logistic regression and random forest classifier.

- **step**: This is used to specify how many features to be removed at every iteration. We used step = 5.

- **scoring**: The metric used to evaluate the performance at every iteration is passed using this parameter. We used accuracy as the scoring metric.

- **cv**: This is used to determine the cross-validation splitting strategy. We set cv = 5 for our model.

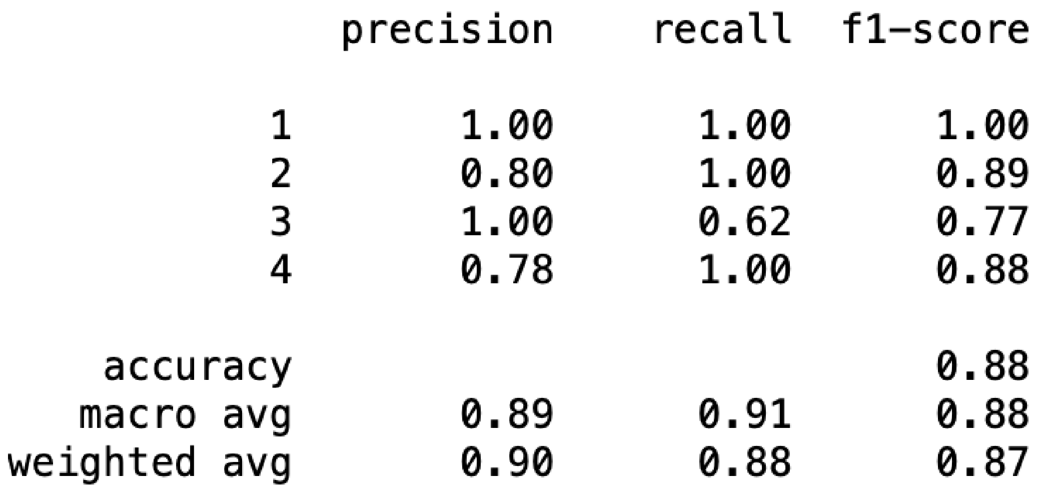
- **min_features_to_select:** This is used to specify the minimum number of features that need to be selected. We set this parameter to the value 800 for our model.

As mentioned in the model details, we used Support Vector Machines, Logistic regression and Random forest classifier as our models. We ran the model for two different values of train-test splits i.e., 75-25 split and 80-20 split.

For 75%-25% train-test split:

For Logistic Regression:

Accuracy: 87.5%



```
              precision    recall  f1-score

           1       1.00      1.00      1.00
           2       0.80      1.00      0.89
           3       1.00      0.62      0.77
           4       0.78      1.00      0.88

    accuracy                           0.88
   macro avg       0.89      0.91      0.88
weighted avg       0.90      0.88      0.87
```

For Random Forest Classifier:

Accuracy: 41.66%

```
              precision    recall  f1-score

           1       0.38      0.60      0.46
           2       0.36      1.00      0.53
           3       1.00      0.25      0.40
           4       0.33      0.14      0.20

    accuracy                           0.42
   macro avg       0.52      0.50      0.40
weighted avg       0.57      0.42      0.38
```

For 80%-20% train-test split:

For Logistic Regression:

Accuracy: 94.44%

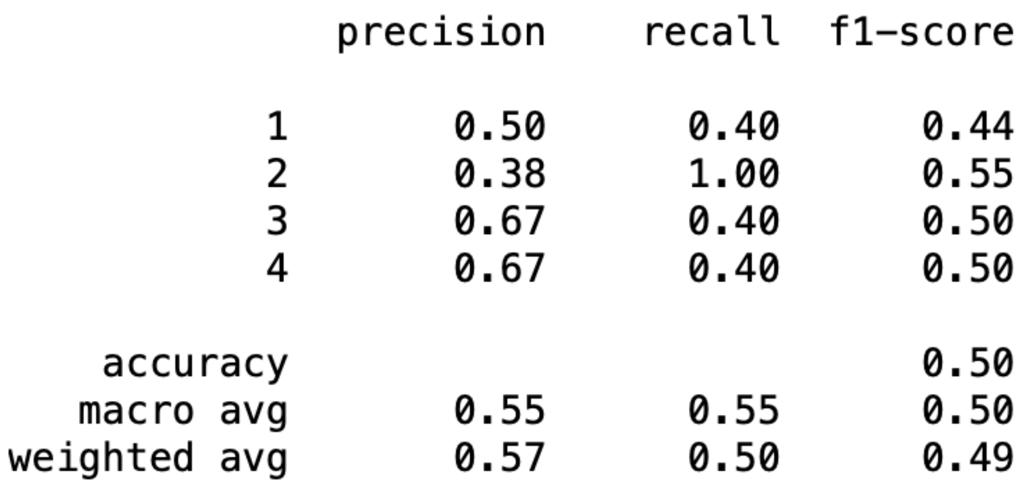
```
              precision    recall  f1-score

           1       1.00      1.00      1.00
           2       0.75      1.00      0.86
           3       1.00      0.80      0.89
           4       1.00      1.00      1.00

    accuracy                           0.94
   macro avg       0.94      0.95      0.94
weighted avg       0.96      0.94      0.95
```

For Random Forest Classifier:

Accuracy: 50%



```
              precision    recall  f1-score

           1       0.50      0.40      0.44
           2       0.38      1.00      0.55
           3       0.67      0.40      0.50
           4       0.67      0.40      0.50

    accuracy                           0.50
   macro avg       0.55      0.55      0.50
weighted avg       0.57      0.50      0.49
```

As we can see from the results above we are getting the best results for logistic regression at the moment. We see that random forest classifier does not perform well and we hypothesize this happens because of the input being numerical data. It gives much better performance when the data is categorical and when there exist some outliers in data. The other reason for it to not perform the way we thought it would, can be the number of samples. If the number of samples would have been more than 90, the random forest classifier would have worked better. Also, we see that the SVM classifier performs almost as well as the Logistic regression model and we thought the same would happen when we use SVM for a classification task with a linear kernel.

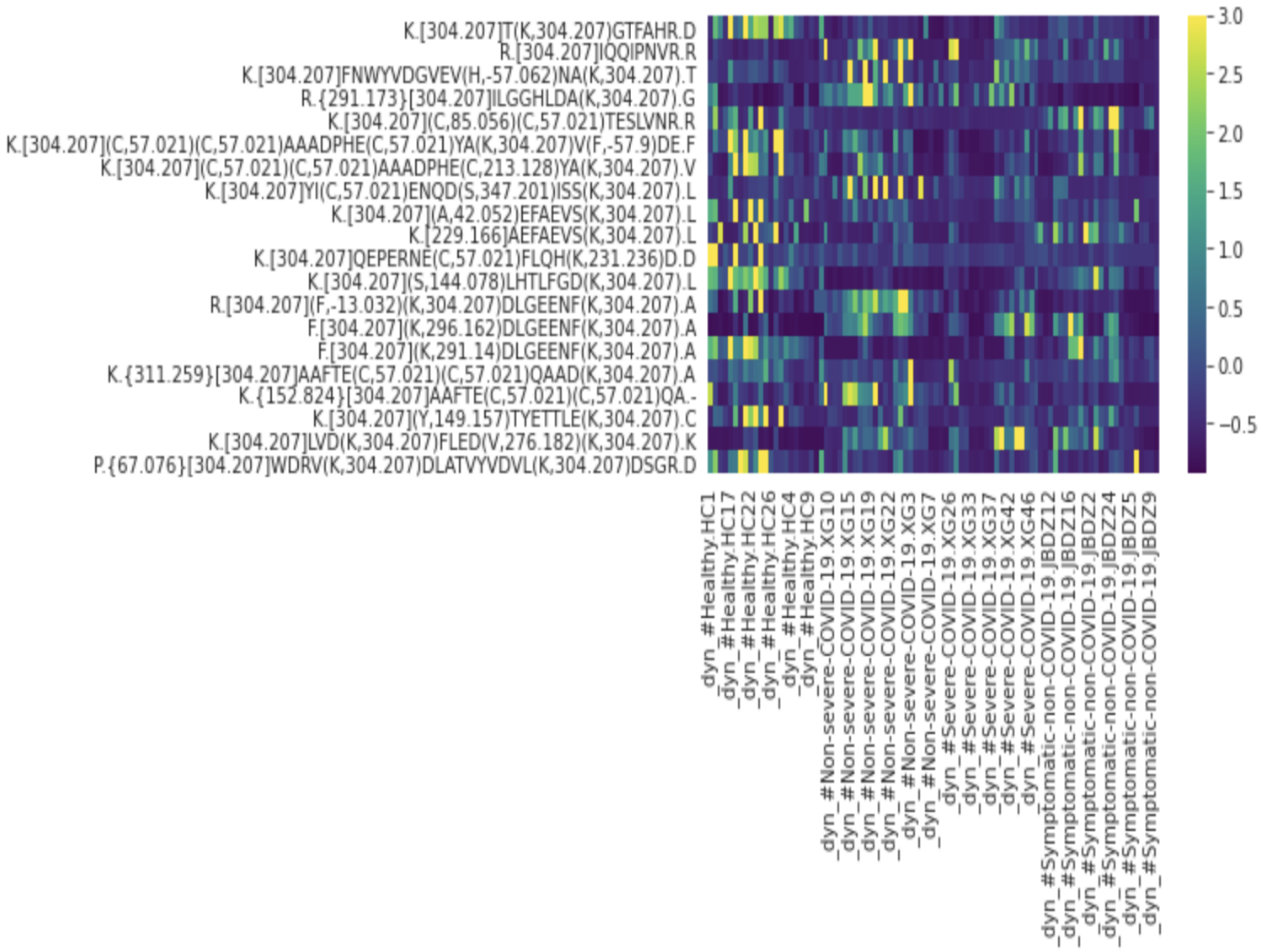
As mentioned in the project plan, our main focus is to reduce the false negatives as a patient who is COVID positive cannot be treated as a healthy patient which is why the objective is to increase the recall. So, we can see that the scores for recall, F1-score and accuracy are best for Logistic regression.

Assessment of important features for the classification

As mentioned in our preliminary report, we used the RFECV feature selection method on the filtered dataset of 945 features. RFECV chose 809, 934 and 800 as the optimal number of features using cross-validation for each of the logistic regression, random forest and SVM classifier models respectively.
We used the SelectKBest function of the sklearn library to choose the top 20 most important peptides. The chosen peptides were:

1. K.[304.207]T(K,304.207)GTFAHR.D
2. R.[304.207]IQQIPNVR.R
3. K.[304.207]FNWYVDGVEV(H,-57.062)NA(K,304.207).T
4. R.{291.173}[304.207]ILGGHLDA(K,304.207).G
5. K.[304.207](C,85.056)(C,57.021)TESLVNR.R
6. K.[304.207](C,57.021)(C,57.021)AAADPHE(C,57.021)YA(K,304.207)V(F,-57.9)DE.F
7. K.[304.207](C,57.021)(C,57.021)AAADPHE(C,213.128)YA(K,304.207).V
8. K.[304.207]YI(C,57.021)ENQD(S,347.201)ISS(K,304.207).L
9. K.[304.207](A,42.052)EFAEVS(K,304.207).L
10. K.[229.166]AEFAEVS(K,304.207).L
11. K.[304.207]QEPERNE(C,57.021)FLQH(K,231.236)D.D
12. K.[304.207](S,144.078)LHTLFGD(K,304.207).L
13. R.[304.207](F,-13.032)(K,304.207)DLGEENF(K,304.207).A
14. F.[304.207](K,296.162)DLGEENF(K,304.207).A
15. F.[304.207](K,291.14)DLGEENF(K,304.207).A
16. K.{311.259}[304.207]AAFTE(C,57.021)(C,57.021)QAAD(K,304.207).A
17. K.{152.824}[304.207]AAFTE(C,57.021)(C,57.021)QA.-
18. K.[304.207](Y,149.157)TYETTLE(K,304.207).C
19. K.[304.207]LVD(K,304.207)FLED(V,276.182)(K,304.207).K
20. P.{67.076}[304.207]WDRV(K,304.207)DLATVYVDVL(K,304.207)DSGR.D

The correlation of these peptides with the each of the four classification labels is captured in the heatmap shown below:



We make a few key observations from the above heatmap:

- The peptides K.[304.207]T(K,304.207)GTFAHR.D, K.[304.207](C,57.021)(C,57.021)AAADPHE(C,57.021)YA(K,304.207)V(F,-57.9)DE.F, K.[304.207](C,57.021)(C,57.021)AAADPHE(C,213.128)YA(K,304.207).V and K.[304.207](Y,149.157)TYETTLE(K,304.207).C (peptides 1, 6, 7 and 18 from the above list respectively) have high correlation with the healthy class and low correlation with all other classes so it might be fair to say that these peptides are likely to be present in healthy individuals.

- Following a similar reasoning, it could be said that the peptide K.[304.207]FNWYVDGVEV(H,-57.062)NA(K,304.207).T (peptide 3) is likely to be present in COVID-19 patients with non-severe symptoms.

- K.[304.207](C,85.056)(C,57.021)TESLVNR.R and R.{291.173}[304.207]ILGGHLDA(K,304.207).G (peptides 5 and 4 respectively) seem to be present in either healthy individuals or those who are symptomatic without COVID-19, with the former having a higher correlation with healthy individuals and the latter with individuals who are symptomatic without COVID-19.

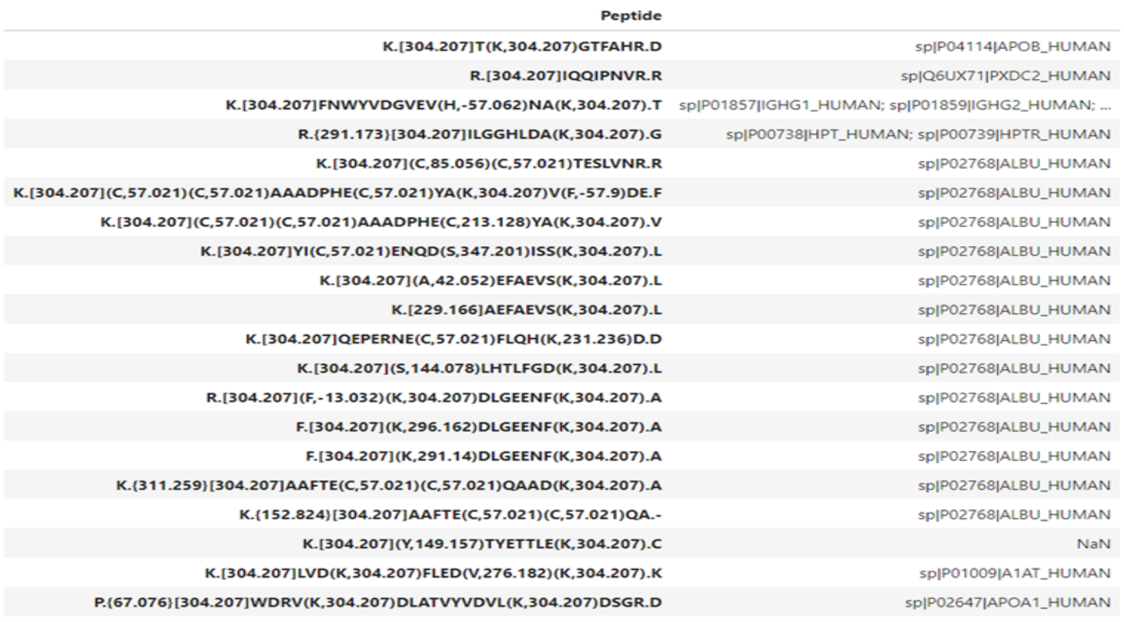
- The peptides R.[304.207]IQQIPNVR.R and K.[304.207]FNWYVDGVEV(H,-57.062)NA(K,304.207).T (peptides 2 and 3 respectively) seem to be present in COVID-19 patients, with the former being highly correlated with both the severe and non-severe cases whereas the latter mostly has a high correlation with the non-severe cases.

- K.[304.207]LVD(K,304.207)FLED(V,276.182)(K,304.207).K (peptides 19) mostly seems to be associated with the severe COVID-19 patient class.

- K.[304.207]YI(C,57.021)ENQD(S,347.201)ISS(K,304.207).L (peptide 8) has a high correlation with the severe COVID-19 class and a low correlation with all the others and so could be used as an indicator of the severe onset of the disease.

Assessment on protein identification

We use the SelectKBest model from sklearn.feature_selection library to select the top 20 peptides. Then, next we find out the top canonical proteins (unique 19 count) associated with these peptides from the data given.4



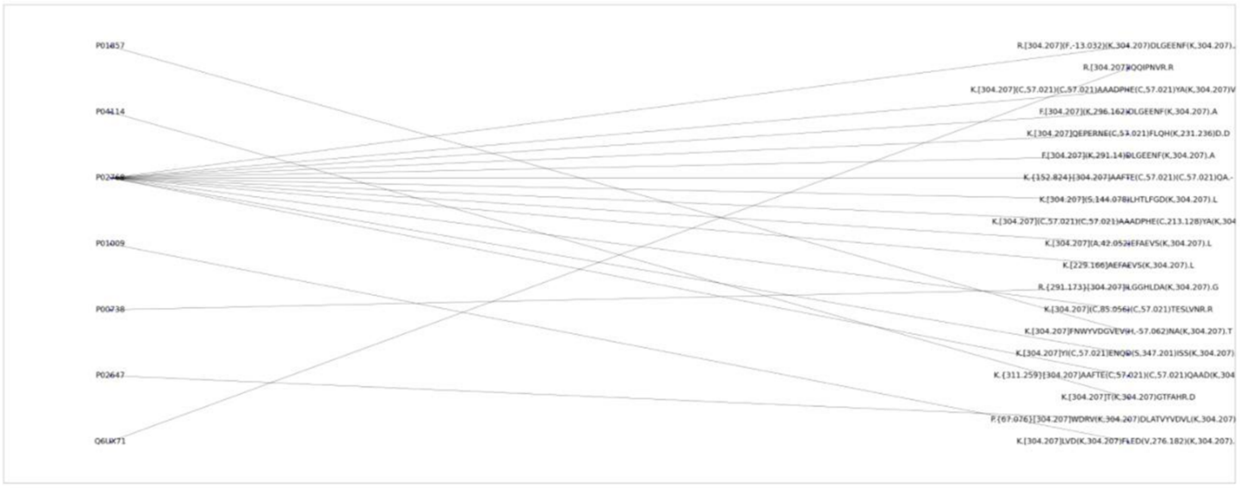
Peptide	
K.[304.207]T(K,304.207)GTFAHR.D	sp|P04114|APOB_HUMAN
R.[304.207]IQQIPNVR.R	sp|Q6UX71|PXDC2_HUMAN
K.[304.207]FNWYVDGVEV(H,-57.062)NA(K,304.207).T	sp|P01857|IGHG1_HUMAN; sp|P01859|IGHG2_HUMAN; ...
R.{291.173}[304.207]ILGGHLDA(K,304.207).G	sp|P00738|HPT_HUMAN; sp|P00739|HPTR_HUMAN
K.[304.207](C,85.056)(C,57.021)TESLVNR.R	sp|P02768|ALBU_HUMAN
K.[304.207](C,57.021)(C,57.021)AAADPHE(C,57.021)YA(K,304.207)V(F,-57.9)DE.F	sp|P02768|ALBU_HUMAN
K.[304.207](C,57.021)(C,57.021)AAADPHE(C,213.128)YA(K,304.207).V	sp|P02768|ALBU_HUMAN
K.[304.207]YI(C,57.021)ENQD(S,347.201)ISS(K,304.207).L	sp|P02768|ALBU_HUMAN
K.[304.207](A,42.052)EFAEVS(K,304.207).L	sp|P02768|ALBU_HUMAN
K.[229.166]AEFAEVS(K,304.207).L	sp|P02768|ALBU_HUMAN
K.[304.207]QEPERNE(C,57.021)FLQH(K,231.236)D.D	sp|P02768|ALBU_HUMAN
K.[304.207](S,144.078)LHTLFGD(K,304.207).L	sp|P02768|ALBU_HUMAN
R.[304.207](F,-13.032)(K,304.207)DLGEENF(K,304.207).A	sp|P02768|ALBU_HUMAN
F.[304.207](K,296.162)DLGEENF(K,304.207).A	sp|P02768|ALBU_HUMAN
F.[304.207](K,291.14)DLGEENF(K,304.207).A	sp|P02768|ALBU_HUMAN
K.{311.259}[304.207]AAFTE(C,57.021)(C,57.021)QAAD(K,304.207).A	sp|P02768|ALBU_HUMAN
K.{152.824}[304.207]AAFTE(C,57.021)(C,57.021)QA.-	sp|P02768|ALBU_HUMAN
K.[304.207](Y,149.157)TYETTLE(K,304.207).C	NaN
K.[304.207]LVD(K,304.207)FLED(V,276.182)(K,304.207).K	sp|P01009|A1AT_HUMAN
P.{67.076}[304.207]WDRV(K,304.207)DLATVYVDVL(K,304.207)DSGR.D	sp|P02647|APOA1_HUMAN

Using the maximum parsimony algorithm, we find a subset of the top proteins. The algorithm has been implemented as follows:

- Initially, we have a many-to-many mapping from the proteins to peptides. From this mapping, we identify the unique one-to-one protein to peptide mapping. In this step, we find a total of 4 mappings, which implies 4 proteins.
- After this, we use the greedy approach in which we select the protein which maps to the maximum number of peptides from the remaining proteins (that is, from the remaining 15 proteins and their mappings). Here, we get a total of 3 proteins.

- In total, we found 7 proteins and we found that all these 7 proteins essentially map to all the top 20 peptides (that is, covering all our peptides).

Protein	Peptide
Q6UX71	['R.[304.207]IQQIPNVR.R']
P01857	['K.[304.207]FNWYVDGVEV(H,-57.062)NA(K,304.207).T']
P00738	['R.{291.173}[304.207]ILGGHLDA(K,304.207).G']
P01009	['K.[304.207]LVD(K,304.207)FLED(V,276.182)(K,304.207).K']
P02768	['K.[304.207](C,85.056)(C,57.021)TESLVNR.R', 'K.[304.207](C,57.021)(C,57.021)AAADPHE(C,57.021)YA(K,304.207)V(F,-57.9)DE.F', 'K.[304.207](C,57.021)(C,57.021)AAADPHE(C,213.128)YA(K,304.207).V', 'K.[304.207]YI(C,57.021)ENQD(S,347.201)ISS(K,304.207).L', 'K.[304.207](A,42.052)EFAEVS(K,304.207).L', 'K.[229.166]AEFAEVS(K,304.207).L', 'K.[304.207]QEPERNE(C,57.021)FLQH(K,231.236)D.D', 'K.[304.207](S,144.078)LHTLFGD(K,304.207).L', 'R.[304.207](F,-13.032)(K,304.207)DLGEENF(K,304.207).A', 'F.[304.207](K,296.162)DLGEENF(K,304.207).A', 'F.[304.207](K,291.14)DLGEENF(K,304.207).A', 'K.{311.259}[304.207]AAFTE(C,57.021)(C,57.021)QAAD(K,304.207).A', 'K.{152.824}[304.207]AAFTE(C,57.021)(C,57.021)QA.-']
P04114	['K.[304.207]T(K,304.207)GTFAHR.D']
P02647	['P.{67.076}[304.207]WDRV(K,304.207)DLATVYVDVL(K,304.207)DSGR.D']

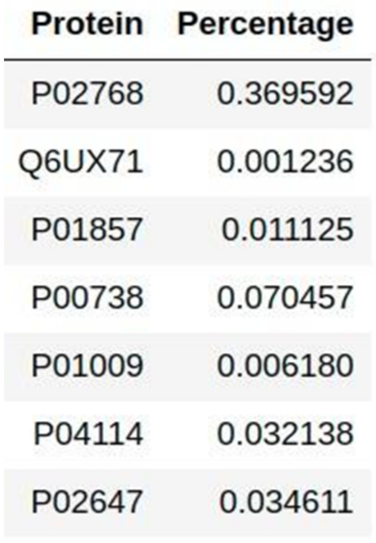

P01857
P04114
P02768
P01009
P00738
P02647
Q6UX71
R.[304.207](F,-13.032)(K,304.207)DLGEENF(K,304.207).
R.[304.207]IQQIPNVR.R
K.[304.207](C,57.021)(C,57.021)AAADPHE(C,57.021)YA(K,304.207)V
F.[304.207](K,296.162)DLGEENF(K,304.207).A
K.[304.207]QEPERNE(C,57.021)FLQH(K,231.236)D.D
F.[304.207](K,291.14)DLGEENF(K,304.207).A
K.{152.824}[304.207]AAFTE(C,57.021)(C,57.021)QA.-
K.[304.207](S,144.078)LHTLFGD(K,304.207).L
K.[304.207](C,57.021)(C,57.021)AAADPHE(C,213.128)YA(K,304
K.[304.207](A,42.052)EFAEVS(K,304.207).L
K.[229.166]AEFAEVS(K,304.207).L
R.{291.173}[304.207]ILGGHLDA(K,304.207).G
K.[304.207](C,85.056)(C,57.021)TESLVNR.R
K.[304.207]FNWYVDGVEV(H,-57.062)NA(K,304.207).T
K.[304.207]YI(C,57.021)ENQD(S,347.201)ISS(K,304.207)
K.{311.259}[304.207]AAFTE(C,57.021)(C,57.021)QAAD(K,304
K.[304.207]T(K,304.207)GTFAHR.D
P.{67.076}[304.207]WDRV(K,304.207)DLATVYVDVL(K,304.207)
K.[304.207]LVD(K,304.207)FLED(V,276.182)(K,304.207).

Assessment on protein level differential abundance

Once we have the top 7 proteins, for each protein, we create a list of peptides which the given protein maps to from the whole dataset, i.e. 100000 features approximately. Following this, we get a total of 7 lists of peptides for each of the proteins. Then, for each list of peptides, we determine the percentage of peptides that are common with the final-feature list (which are the 809 selected peptides). That is,

$$Protein\ level\ abundance\ (\%)\ = \frac{len\ (peptide\ list\ p_i \cap final-feature\ list)}{len\ (final-feature\ list) = 809} \quad \exists\ i\ \in\ \{1,\ 2,\ \dots,\ 7\}$$

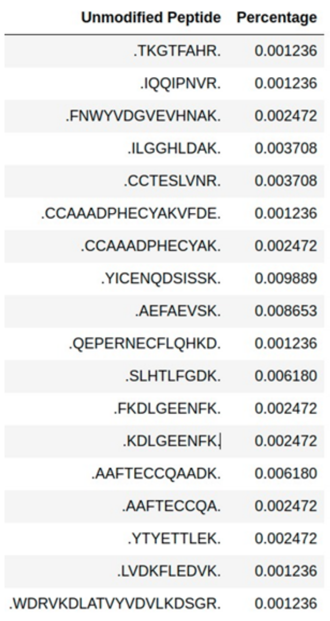
We get the following table that shows the percentage for each of the 7 peptides.



Protein	Percentage
P02768	0.369592
Q6UX71	0.001236
P01857	0.011125
P00738	0.070457
P01009	0.006180
P04114	0.032138
P02647	0.034611

We find that none of the protein-level abundance was higher than the value of 0.5. Therefore, according to the majority rule, which states that the effect is considered to be protein-level if a majority of the peptides mapping uniquely to the protein is significantly different between the classes; none of the proteins passed the majority rule. Therefore, we hypothesize that the observed differential abundancy effect is peptide-level effect and not the protein-level effect.

Assessment on peptide/variant level differential abundance



Unmodified Peptide	Percentage
.TKGTFAHR.	0.001236
.IQQIPNVR.	0.001236
.FNWYVDGVEVHNAK.	0.002472
.ILGGHLDAK.	0.003708
.CCTESLVNR.	0.003708
.CCAAADPHECYAKVFDE.	0.001236
.CCAAADPHECYAK.	0.002472
.YICENQDSISSK.	0.009889
.AEFAEVSK.	0.008653
.QEPERNECFLQHKD.	0.001236
.SLHTLFGDK.	0.006180
.FKDLGEENFK.	0.002472
.KDLGEENFK.	0.002472
.AAFTECCQAADK.	0.006180
.AAFTECCQA.	0.002472
.YTYETTLEK.	0.002472
.LVDKFLEDVK.	0.001236
.WDRVKDLATVYVDVLKDSGR.	0.001236

We perform a similar analysis as we did to find out if the effect was protein level or peptide, on the unmodified peptide column to assess whether the analysis is peptide level or variant level. The table above shows that none of the unmodified peptide abundance was higher than the value of 0.5. Therefore, according to the majority rule, which states that the effect is considered to be peptide-level if a majority of the unmodified peptides mapping uniquely to the protein is significantly different between the classes; none of the peptides passed the majority rule. Therefore, we hypothesize that the observed differential abundancy effect is variant-level effect and not the peptide-level effect.