SCEM Final Coursework

Task II

Rohan Anthony (2704500)

IMPORTANT NOTES:

- **DO NOT** change the code block names. Enter your solutions to each question into the predefined code blocks.
- **Don't forget to replace the placeholder value in code block "Setup" by your valid student ID number.**
- If you want to use any packages, add their names to the vector `required.packages` in the code block "Setup". That code block will automatically install those packages (if needed) and load them. Any other calls to `install.packages()`, `library()` or `require()` may result in zero marks for the activity where it happens.
- Make sure that the data files *df.rds* and *df_holdout.rds* are located in the same folder as this file, *Task02.Rmd*.
- Instructions for this task are provided in blue text below. Your solution should NOT be added within those text boxes, and should NOT be in blue text. **During marking, all the text within the text blocks below, with names starting with "prompt", will be removed and not visible to the marker.**

```
##
## List of allowed packages (pre-loaded):
##   tidyverse
##   tidymodels
##   recipes
##   parsnip
##   bonsai
##   healthyR.ai
##   knitr
##   ranger
```

In this task you will build a full predictive pipeline for the problem described in the coursework specs. Please read each activity carefully and follow the instructions to complete your coursework.

Make sure that the data file **df.rds** is placed in the same folder as this Task02.Rmd file. The data that you can use for model development in this task will then be automatically loaded and set up for you, and stored into a dataframe variable called `df`.

---

1. Exploratory Data Analysis

Note: It is expected that this section should result in no more than 600-ish words (possibly less; not counting the code blocks).

Add here your exploratory data analysis. Your solution should include both your code (which markers must be able to run independently) and your rationale for each exploratory step. In general, your EDA steps should include:

- A short justification of what a given plot or statistical summary is intended to reveal about the data.
- A code block implementing your EDA step.

At the end of your EDA section, you should also include a short commentary on what your data exploration revealed about the data, and which pre-processing steps may be required based on that.

You can use as many code blocks as needed, but don't overdo it (remember - everything needs to have a proper rationale).

```
# EDA Step 1: Overview of data set, structure and size
cat("Observations (Rows):", nrow(df), "\n")
## Observations (Rows): 7403
cat("Variables (Columns):", ncol(df), "\n")
## Variables (Columns): 390

glimpse(df)
## Rows: 7,403
## Columns: 390
## $ Info_PepID                              <chr> "AAC06355.1:6", "NP_040304.1:6"~
## $ Info_group                              <int> 156, 87, 7, 100, 620, 70, 55, 5~
## $ Info_protein_id                         <chr> "AAC06355.1", "NP_040304.1", "A~
## $ Info_pos                                <dbl> 63.0000, 477.5000, 114.5000, 65~
## $ feat_local_Entropy                      <dbl> 3.154677, 2.898951, 3.179300, 2~
## $ feat_local_MolWeight                    <dbl> 1873.968, 1870.371, 2033.281, 2~
## $ feat_local_AAtypes_Tiny                 <dbl> 0.3703704, 0.4366667, 0.2666667~
## $ feat_local_AAtypes_Small                <dbl> 0.5555556, 0.5800000, 0.6190476~
## $ feat_local_AAtypes_Aliphatic            <dbl> 0.2740741, 0.2700000, 0.1952381~
## $ feat_local_AAtypes_Aromatic             <dbl> 0.06666667, 0.04000000, 0.22857~
## $ feat_local_AAtypes_NonPolar             <dbl> 0.5629630, 0.4800000, 0.6000000~
## $ feat_local_AAtypes_Polar                <dbl> 0.4370370, 0.5200000, 0.4000000~
## $ feat_local_AAtypes_Charged              <dbl> 0.10370370, 0.28000000, 0.21428~
## $ feat_local_AAtypes_Basic                <dbl> 0.03703704, 0.28000000, 0.08571~
## $ feat_local_AAtypes_Acidic               <dbl> 0.06666667, 0.00000000, 0.12857~
## $ feat_local_Atoms_nC                     <dbl> 69.33333, 68.75000, 80.21429, 7~
## $ feat_local_Atoms_nH                     <dbl> 135.5556, 149.9000, 141.1429, 1~
## $ feat_local_Atoms_nN                     <dbl> 21.55556, 24.90000, 18.57143, 2~
## $ feat_local_Atoms_nO                     <dbl> 37.00000, 34.05000, 39.85714, 4~
## $ feat_local_Atoms_nS                     <dbl> 0.3333333, 0.0000000, 0.9285714~
## $ feat_local_AAC_A                        <dbl> 0.15555556, 0.10333333, 0.07619~
## $ feat_local_AAC_R                        <dbl> 0.03703704, 0.18000000, 0.04761~
## $ feat_local_AAC_N                        <dbl> 0.118518519, 0.000000000, 0.057~
## $ feat_local_AAC_D                        <dbl> 0.000000000, 0.000000000, 0.104~
## $ feat_local_AAC_C                        <dbl> 0.000000000, 0.000000000, 0.061~
## $ feat_local_AAC_E                        <dbl> 0.066666667, 0.000000000, 0.023~
## $ feat_local_AAC_Q                        <dbl> 0.140740741, 0.020000000, 0.000~
## $ feat_local_AAC_G                        <dbl> 0.14074074, 0.11333333, 0.00000~
## $ feat_local_AAC_H                        <dbl> 0.000000000, 0.000000000, 0.000~
## $ feat_local_AAC_I                        <dbl> 0.04444444, 0.05000000, 0.00000~
## $ feat_local_AAC_L                        <dbl> 0.066666667, 0.030000000, 0.042~
## $ feat_local_AAC_K                        <dbl> 0.000000000, 0.100000000, 0.038~
## $ feat_local_AAC_M                        <dbl> 0.02222222, 0.00000000, 0.00000~
## $ feat_local_AAC_F                        <dbl> 0.000000000, 0.010000000, 0.014~
## $ feat_local_AAC_P                        <dbl> 0.059259259, 0.056666667, 0.114~
## $ feat_local_AAC_S                        <dbl> 0.000000000, 0.170000000, 0.128~
## $ feat_local_AAC_T                        <dbl> 0.074074074, 0.050000000, 0.000~
## $ feat_local_AAC_W                        <dbl> 0.066666667, 0.000000000, 0.000~
## $ feat_local_AAC_Y                        <dbl> 0.00000000, 0.03000000, 0.21428~
## $ feat_local_AAC_V                        <dbl> 0.007407407, 0.086666667, 0.076~
## $ feat_local_CTDC_hydrophobicity.Group1   <dbl> 0.3629630, 0.3000000, 0.2714286~
## $ feat_local_CTDC_hydrophobicity.Group2   <dbl> 0.4296296, 0.5233333, 0.5333333~
## $ feat_local_CTDC_hydrophobicity.Group3   <dbl> 0.2074074, 0.1766667, 0.1952381~
## $ feat_local_CTDC_normwaalsvolume.Group1 <dbl> 0.4296296, 0.4933333, 0.4857143~
## $ feat_local_CTDC_normwaalsvolume.Group2 <dbl> 0.4444444, 0.1866667, 0.2000000~
## $ feat_local_CTDC_normwaalsvolume.Group3 <dbl> 0.1259259, 0.3200000, 0.3142857~
## $ feat_local_CTDC_polarity.Group1         <dbl> 0.2074074, 0.2066667, 0.4095238~
## $ feat_local_CTDC_polarity.Group2         <dbl> 0.4296296, 0.4933333, 0.3190476~
## $ feat_local_CTDC_polarity.Group3         <dbl> 0.3629630, 0.3000000, 0.2714286~
## $ feat_local_CTDC_polarizability.Group1   <dbl> 0.3703704, 0.4366667, 0.3095238~
## $ feat_local_CTDC_polarizability.Group2   <dbl> 0.5037037, 0.2433333, 0.3761905~
## $ feat_local_CTDC_polarizability.Group3   <dbl> 0.1259259, 0.3200000, 0.3142857~
## $ feat_local_CTDC_charge.Group1           <dbl> 0.037037037, 0.280000000, 0.085~
## $ feat_local_CTDC_charge.Group2           <dbl> 0.8962963, 0.7200000, 0.7857143~
## $ feat_local_CTDC_charge.Group3           <dbl> 0.06666667, 0.00000000, 0.12857~
## $ feat_local_CTDC_secondarystruct.Group1 <dbl> 0.4888889, 0.4333333, 0.2285714~
## $ feat_local_CTDC_secondarystruct.Group2 <dbl> 0.1925926, 0.2266667, 0.3666667~
## $ feat_local_CTDC_secondarystruct.Group3 <dbl> 0.3185185, 0.3400000, 0.4047619~
## $ feat_local_CTDC_solventaccess.Group1   <dbl> 0.4814815, 0.3933333, 0.2714286~
## $ feat_local_CTDC_solventaccess.Group2   <dbl> 0.3629630, 0.3000000, 0.2714286~
## $ feat_local_CTDC_solventaccess.Group3   <dbl> 0.15555556, 0.30666667, 0.45714~
## $ feat_local_CTDD_prop1.G1.residue0      <dbl> 11.85185, 15.00000, 16.19048, 1~
## $ feat_local_CTDD_prop1.G1.residue25     <dbl> 11.85185, 15.00000, 16.19048, 2~
## $ feat_local_CTDD_prop1.G1.residue50     <dbl> 28.88889, 35.33333, 29.52381, 4~
## $ feat_local_CTDD_prop1.G1.residue75     <dbl> 44.44444, 55.66667, 50.95238, 6~
## $ feat_local_CTDD_prop1.G1.residue100    <dbl> 74.81481, 88.66667, 87.61905, 8~
## $ feat_local_CTDD_prop1.G2.residue0      <dbl> 21.48148, 17.00000, 18.57143, 2~
## $ feat_local_CTDD_prop1.G2.residue25     <dbl> 32.59259, 24.33333, 25.71429, 2~
## $ feat_local_CTDD_prop1.G2.residue50     <dbl> 62.96296, 48.66667, 47.14286, 3~
## $ feat_local_CTDD_prop1.G2.residue75     <dbl> 80.74074, 71.33333, 70.47619, 5~
## $ feat_local_CTDD_prop1.G2.residue100    <dbl> 94.81481, 94.33333, 95.23810, 8~
## $ feat_local_CTDD_prop1.G3.residue0      <dbl> 13.33333, 26.00000, 26.19048, 2~
## $ feat_local_CTDD_prop1.G3.residue25     <dbl> 13.33333, 26.00000, 26.19048, 2~
## $ feat_local_CTDD_prop1.G3.residue50     <dbl> 20.00000, 27.00000, 32.38095, 3~
## $ feat_local_CTDD_prop1.G3.residue75     <dbl> 37.77778, 39.66667, 49.52381, 6~
## $ feat_local_CTDD_prop1.G3.residue100    <dbl> 79.25926, 76.33333, 87.14286, 8~
## $ feat_local_CTDD_prop2.G1.residue0      <dbl> 21.48148, 17.66667, 15.71429, 1~
## $ feat_local_CTDD_prop2.G1.residue25     <dbl> 32.59259, 22.66667, 23.33333, 1~
## $ feat_local_CTDD_prop2.G1.residue50     <dbl> 62.96296, 48.66667, 50.95238, 4~
## $ feat_local_CTDD_prop2.G1.residue75     <dbl> 80.74074, 71.33333, 71.42857, 6~
## $ feat_local_CTDD_prop2.G1.residue100    <dbl> 94.81481, 94.33333, 95.71429, 9~
## $ feat_local_CTDD_prop2.G2.residue0      <dbl> 9.62963, 24.00000, 21.90476, 17~
## $ feat_local_CTDD_prop2.G2.residue25     <dbl> 14.07407, 24.00000, 21.90476, 1~
## $ feat_local_CTDD_prop2.G2.residue50     <dbl> 26.66667, 25.66667, 27.14286, 4~
## $ feat_local_CTDD_prop2.G2.residue75     <dbl> 40.74074, 41.00000, 48.09524, 6~
## $ feat_local_CTDD_prop2.G2.residue100    <dbl> 61.48148, 76.33333, 85.71429, 8~
## $ feat_local_CTDD_prop2.G3.residue0      <dbl> 46.66667, 16.00000, 13.33333, 3~
## $ feat_local_CTDD_prop2.G3.residue25     <dbl> 46.66667, 16.00000, 13.33333, 3~
## $ feat_local_CTDD_prop2.G3.residue50     <dbl> 46.66667, 36.33333, 31.90476, 3~
## $ feat_local_CTDD_prop2.G3.residue75     <dbl> 55.55556, 56.00000, 52.85714, 4~
## $ feat_local_CTDD_prop2.G3.residue100    <dbl> 88.14815, 88.66667, 89.04762, 7~
## $ feat_local_CTDD_prop3.G1.residue0      <dbl> 13.33333, 18.00000, 15.71429, 2~
## $ feat_local_CTDD_prop3.G1.residue25     <dbl> 13.33333, 18.00000, 15.71429, 2~
## $ feat_local_CTDD_prop3.G1.residue50     <dbl> 20.00000, 20.66667, 41.90476, 3~
## $ feat_local_CTDD_prop3.G1.residue75     <dbl> 37.77778, 41.00000, 63.80952, 6~
## $ feat_local_CTDD_prop3.G1.residue100    <dbl> 79.25926, 76.33333, 94.76190, 8~
## $ feat_local_CTDD_prop3.G2.residue0      <dbl> 21.48148, 17.66667, 20.47619, 2~
## $ feat_local_CTDD_prop3.G2.residue25     <dbl> 32.59259, 22.66667, 20.47619, 2~
## $ feat_local_CTDD_prop3.G2.residue50     <dbl> 62.96296, 48.66667, 40.47619, 3~
## $ feat_local_CTDD_prop3.G2.residue75     <dbl> 80.74074, 71.33333, 59.04762, 5~
## $ feat_local_CTDD_prop3.G2.residue100    <dbl> 94.81481, 94.33333, 92.85714, 8~
## $ feat_local_CTDD_prop3.G3.residue0      <dbl> 11.85185, 15.00000, 16.19048, 1~
## $ feat_local_CTDD_prop3.G3.residue25     <dbl> 11.85185, 15.00000, 16.19048, 2~
## $ feat_local_CTDD_prop3.G3.residue50     <dbl> 28.88889, 35.33333, 29.52381, 4~
## $ feat_local_CTDD_prop3.G3.residue75     <dbl> 44.44444, 55.66667, 50.95238, 6~
## $ feat_local_CTDD_prop3.G3.residue100    <dbl> 74.81481, 88.66667, 87.61905, 8~
## $ feat_local_CTDD_prop4.G1.residue0      <dbl> 21.48148, 18.33333, 20.47619, 1~
## $ feat_local_CTDD_prop4.G1.residue25     <dbl> 21.48148, 19.00000, 20.47619, 1~
## $ feat_local_CTDD_prop4.G1.residue50     <dbl> 58.51852, 46.66667, 42.85714, 3~
## $ feat_local_CTDD_prop4.G1.residue75     <dbl> 79.25926, 66.66667, 59.52381, 5~
## $ feat_local_CTDD_prop4.G1.residue100    <dbl> 93.33333, 91.66667, 90.95238, 8~
## $ feat_local_CTDD_prop4.G2.residue0      <dbl> 9.62963, 23.33333, 18.09524, 14~
## $ feat_local_CTDD_prop4.G2.residue25     <dbl> 14.81481, 23.33333, 18.09524, 1~
## $ feat_local_CTDD_prop4.G2.residue50     <dbl> 29.62963, 36.00000, 39.52381, 4~
## $ feat_local_CTDD_prop4.G2.residue75     <dbl> 43.70370, 55.66667, 61.90476, 6~
## $ feat_local_CTDD_prop4.G2.residue100    <dbl> 82.22222, 87.66667, 92.38095, 9~
## $ feat_local_CTDD_prop4.G3.residue0      <dbl> 46.66667, 16.00000, 13.33333, 3~
## $ feat_local_CTDD_prop4.G3.residue25     <dbl> 46.66667, 16.00000, 13.33333, 3~
## $ feat_local_CTDD_prop4.G3.residue50     <dbl> 46.66667, 36.33333, 31.90476, 3~
## $ feat_local_CTDD_prop4.G3.residue75     <dbl> 55.55556, 56.00000, 52.85714, 4~
## $ feat_local_CTDD_prop4.G3.residue100    <dbl> 88.14815, 88.66667, 89.04762, 7~
## $ feat_local_CTDD_prop5.G1.residue0      <dbl> 31.11111, 20.33333, 37.14286, 1~
## $ feat_local_CTDD_prop5.G1.residue25     <dbl> 31.11111, 20.33333, 37.14286, 1~
## $ feat_local_CTDD_prop5.G1.residue50     <dbl> 31.11111, 37.33333, 37.14286, 2~
## $ feat_local_CTDD_prop5.G1.residue75     <dbl> 31.11111, 59.00000, 37.14286, 3~
## $ feat_local_CTDD_prop5.G1.residue100    <dbl> 32.59259, 88.66667, 42.85714, 4~
## $ feat_local_CTDD_prop5.G2.residue0      <dbl> 8.888889, 9.333333, 9.047619, 1~
## $ feat_local_CTDD_prop5.G2.residue25     <dbl> 25.18519, 19.33333, 22.38095, 2~
## $ feat_local_CTDD_prop5.G2.residue50     <dbl> 48.88889, 45.66667, 49.04762, 4~
## $ feat_local_CTDD_prop5.G2.residue75     <dbl> 69.62963, 71.00000, 72.38095, 6~
## $ feat_local_CTDD_prop5.G2.residue100    <dbl> 97.77778, 96.33333, 98.57143, 9~
## $ feat_local_CTDD_prop5.G3.residue0      <dbl> 8.148148, 0.000000, 39.523810, ~
## $ feat_local_CTDD_prop5.G3.residue25     <dbl> 8.148148, 0.000000, 39.523810, ~
## $ feat_local_CTDD_prop5.G3.residue50     <dbl> 8.148148, 0.000000, 39.523810, ~
## $ feat_local_CTDD_prop5.G3.residue75     <dbl> 8.148148, 0.000000, 39.523810, ~
## $ feat_local_CTDD_prop5.G3.residue100    <dbl> 11.11111, 0.00000, 71.42857, 78~
## $ feat_local_CTDD_prop6.G1.residue0      <dbl> 11.851852, 13.333333, 32.857143~
## $ feat_local_CTDD_prop6.G1.residue25     <dbl> 15.55556, 15.66667, 32.85714, 1~
## $ feat_local_CTDD_prop6.G1.residue50     <dbl> 40.00000, 42.33333, 39.04762, 3~
## $ feat_local_CTDD_prop6.G1.residue75     <dbl> 67.40741, 65.00000, 47.14286, 6~
## $ feat_local_CTDD_prop6.G1.residue100    <dbl> 95.55556, 95.00000, 68.57143, 9~
## $ feat_local_CTDD_prop6.G2.residue0      <dbl> 17.03704, 19.66667, 16.66667, 3~
## $ feat_local_CTDD_prop6.G2.residue25     <dbl> 17.03704, 19.66667, 16.66667, 3~
## $ feat_local_CTDD_prop6.G2.residue50     <dbl> 22.96296, 35.66667, 40.95238, 4~
## $ feat_local_CTDD_prop6.G2.residue75     <dbl> 31.85185, 52.33333, 55.71429, 4~
## $ feat_local_CTDD_prop6.G2.residue100    <dbl> 79.25926, 77.66667, 87.61905, 6~
## $ feat_local_CTDD_prop6.G3.residue0      <dbl> 25.925926, 18.666667, 18.095238~
## $ feat_local_CTDD_prop6.G3.residue25     <dbl> 25.92593, 18.66667, 18.09524, 1~
## $ feat_local_CTDD_prop6.G3.residue50     <dbl> 49.62963, 38.33333, 47.61905, 3~
## $ feat_local_CTDD_prop6.G3.residue75     <dbl> 67.40741, 56.33333, 66.19048, 6~
## $ feat_local_CTDD_prop6.G3.residue100    <dbl> 92.59259, 91.00000, 94.76190, 8~
## $ feat_local_CTDD_prop7.G1.residue0      <dbl> 20.00000, 13.66667, 16.66667, 1~
## $ feat_local_CTDD_prop7.G1.residue25     <dbl> 33.33333, 13.66667, 16.66667, 1~
## $ feat_local_CTDD_prop7.G1.residue50     <dbl> 60.00000, 40.00000, 35.71429, 3~
## $ feat_local_CTDD_prop7.G1.residue75     <dbl> 79.25926, 61.66667, 60.00000, 6~
## $ feat_local_CTDD_prop7.G1.residue100    <dbl> 97.03704, 92.33333, 90.95238, 8~
## $ feat_local_CTDD_prop7.G2.residue0      <dbl> 11.85185, 15.00000, 16.19048, 1~
## $ feat_local_CTDD_prop7.G2.residue25     <dbl> 11.85185, 15.00000, 16.19048, 2~
## $ feat_local_CTDD_prop7.G2.residue50     <dbl> 28.88889, 35.33333, 29.52381, 4~
## $ feat_local_CTDD_prop7.G2.residue75     <dbl> 44.44444, 55.66667, 50.95238, 6~
## $ feat_local_CTDD_prop7.G2.residue100    <dbl> 74.81481, 88.66667, 87.61905, 8~
## $ feat_local_CTDD_prop7.G3.residue0      <dbl> 20.00000, 20.00000, 19.52381, 3~
## $ feat_local_CTDD_prop7.G3.residue25     <dbl> 20.00000, 20.00000, 21.90476, 3~
## $ feat_local_CTDD_prop7.G3.residue50     <dbl> 20.74074, 37.66667, 45.23810, 4~
## $ feat_local_CTDD_prop7.G3.residue75     <dbl> 28.88889, 59.33333, 68.57143, 5~
## $ feat_local_CTDD_prop7.G3.residue100    <dbl> 74.07407, 90.66667, 94.76190, 7~
## $ feat_local_CTDT_prop1.Tr1221           <dbl> 0.15873016, 0.33928571, 0.32653~
## $ feat_local_CTDT_prop1.Tr1331           <dbl> 0.19047619, 0.12500000, 0.16326~
## $ feat_local_CTDT_prop1.Tr2332           <dbl> 0.21428571, 0.17500000, 0.15306~
## $ feat_local_CTDT_prop2.Tr1221           <dbl> 0.19841270, 0.19642857, 0.26530~
## $ feat_local_CTDT_prop2.Tr1331           <dbl> 0.1746032, 0.3142857, 0.5612245~
## $ feat_local_CTDT_prop2.Tr2332           <dbl> 0.03174603, 0.11071429, 0.07142~
## $ feat_local_CTDT_prop3.Tr1221           <dbl> 0.21428571, 0.20357143, 0.35714~
## $ feat_local_CTDT_prop3.Tr1331           <dbl> 0.19047619, 0.15714286, 0.26530~
## $ feat_local_CTDT_prop3.Tr2332           <dbl> 0.15873016, 0.30714286, 0.22448~
## $ feat_local_CTDT_prop4.Tr1221           <dbl> 0.25396825, 0.25357143, 0.19897~
## $ feat_local_CTDT_prop4.Tr1331           <dbl> 0.11111111, 0.25714286, 0.35714~
## $ feat_local_CTDT_prop4.Tr2332           <dbl> 0.09523810, 0.16785714, 0.27551~
## $ feat_local_CTDT_prop5.Tr1221           <dbl> 0.03174603, 0.42500000, 0.14285~
## $ feat_local_CTDT_prop5.Tr1331           <dbl> 0.000000000, 0.000000000, 0.025~
## $ feat_local_CTDT_prop5.Tr2332           <dbl> 0.06349206, 0.00000000, 0.22959~
## $ feat_local_CTDT_prop6.Tr1221           <dbl> 0.2063492, 0.2392857, 0.1173469~
## $ feat_local_CTDT_prop6.Tr1331           <dbl> 0.4841270, 0.3321429, 0.1530612~
## $ feat_local_CTDT_prop6.Tr2332           <dbl> 0.16666667, 0.21428571, 0.45918~
## $ feat_local_CTDT_prop7.Tr1221           <dbl> 0.26190476, 0.22857143, 0.20408~
## $ feat_local_CTDT_prop7.Tr1331           <dbl> 0.18253968, 0.29285714, 0.26530~
## $ feat_local_CTDT_prop7.Tr2332           <dbl> 0.08730159, 0.23571429, 0.28571~
## $ feat_local_BLOSUM_scl1.lag1            <dbl> -0.0023279869, -0.0021108143, -~
## $ feat_local_BLOSUM_scl2.lag1            <dbl> -0.0099630031, 0.0127144151, -0~
## $ feat_local_BLOSUM_scl3.lag1            <dbl> 0.0111289988, -0.0060870561, 0.~
## $ feat_local_BLOSUM_scl4.lag1            <dbl> 0.0096937337, -0.0075962157, 0.~
## $ feat_local_BLOSUM_scl5.lag1            <dbl> 0.0189491654, -0.0011288491, 0.~
## $ feat_local_BLOSUM_scl1.lag2            <dbl> -0.0074162825, 0.0003518139, -0~
## $ feat_local_BLOSUM_scl2.lag2            <dbl> -0.002698221, 0.011674983, 0.01~
## $ feat_local_BLOSUM_scl3.lag2            <dbl> 0.0253925405, -0.0169140376, 0.~
## $ feat_local_BLOSUM_scl4.lag2            <dbl> 1.076043e-02, 1.501723e-02, 1.7~
## $ feat_local_BLOSUM_scl5.lag2            <dbl> 0.0055571786, 0.0028170214, -0.~
## $ feat_local_BLOSUM_scl1.lag3            <dbl> 0.0091900794, -0.0090019801, 0.~
## $ feat_local_BLOSUM_scl2.lag3            <dbl> 0.0080572460, 0.0109872894, -0.~
## $ feat_local_BLOSUM_scl3.lag3            <dbl> -3.723117e-03, -9.054776e-03, 3~
## $ feat_local_BLOSUM_scl4.lag3            <dbl> 0.006536871, -0.001129384, 0.01~
## $ feat_local_BLOSUM_scl5.lag3            <dbl> 0.0098394346, -0.0026013831, -0~
## $ feat_local_BLOSUM_scl1.2.lag1          <dbl> -0.0121112050, -0.0100678990, -~
## $ feat_local_BLOSUM_scl1.3.lag1          <dbl> 7.779993e-03, 1.475816e-02, 1.7~
## $ feat_local_BLOSUM_scl1.4.lag1          <dbl> -0.0093373478, 0.0053263333, -0~
## $ feat_local_BLOSUM_scl1.5.lag1          <dbl> 0.0130077924, -0.0043230172, 0.~
## $ feat_local_BLOSUM_scl2.3.lag1          <dbl> -0.0104781466, -0.0054288416, 0~
## $ feat_local_BLOSUM_scl2.4.lag1          <dbl> 0.0191564294, 0.0060599810, 0.0~
## $ feat_local_BLOSUM_scl2.5.lag1          <dbl> -0.0156301799, 0.0001262462, -0~
## $ feat_local_BLOSUM_scl3.4.lag1          <dbl> -0.0194428495, 0.0089834895, 0.~
## $ feat_local_BLOSUM_scl3.5.lag1          <dbl> -0.0267405322, 0.0028691614, 0.~
## $ feat_local_BLOSUM_scl4.5.lag1          <dbl> 0.0233155247, 0.0090978621, -0.~
## $ feat_local_BLOSUM_scl2.1.lag1          <dbl> 0.0078733990, -0.0039931470, 0.~
## $ feat_local_BLOSUM_scl3.1.lag1          <dbl> -0.011736921, -0.018992784, -0.~
## $ feat_local_BLOSUM_scl4.1.lag1          <dbl> -4.308242e-03, -2.042638e-03, -~
## $ feat_local_BLOSUM_scl5.1.lag1          <dbl> -0.0012405778, 0.0036746482, -0~
## $ feat_local_BLOSUM_scl3.2.lag1          <dbl> -0.0132592457, 0.0052956016, 0.~
## $ feat_local_BLOSUM_scl4.2.lag1          <dbl> -0.006328516, 0.013984485, -0.0~
## $ feat_local_BLOSUM_scl5.2.lag1          <dbl> 0.0113273454, 0.0017214587, -0.~
## $ feat_local_BLOSUM_scl4.3.lag1          <dbl> -0.019616305, -0.009108499, -0.~
## $ feat_local_BLOSUM_scl5.3.lag1          <dbl> -2.232729e-02, 6.234786e-03, -1~
## $ feat_local_BLOSUM_scl5.4.lag1          <dbl> 0.0034218963, 0.0066402513, -0.~
## $ feat_local_BLOSUM_scl1.2.lag2          <dbl> 5.706679e-03, -5.412644e-03, 5.~
## $ feat_local_BLOSUM_scl1.3.lag2          <dbl> 0.0092683810, 0.0057179566, 0.0~
## $ feat_local_BLOSUM_scl1.4.lag2          <dbl> -0.0145733669, -0.0030249197, -~
## $ feat_local_BLOSUM_scl1.5.lag2          <dbl> -0.013864227, -0.007525669, -0.~
## $ feat_local_BLOSUM_scl2.3.lag2          <dbl> 0.0057506507, 0.0178083661, -0.~
## $ feat_local_BLOSUM_scl2.4.lag2          <dbl> -0.0066177025, 0.0042424229, 0.~
## $ feat_local_BLOSUM_scl2.5.lag2          <dbl> 0.0103530999, -0.0033778493, 0.~
## $ feat_local_BLOSUM_scl3.4.lag2          <dbl> -0.0237110768, -0.0051244662, 0~
## $ feat_local_BLOSUM_scl3.5.lag2          <dbl> -3.054442e-02, 1.063917e-02, -1~
## $ feat_local_BLOSUM_scl4.5.lag2          <dbl> 0.019123383, 0.007414216, -0.00~
## $ feat_local_BLOSUM_scl2.1.lag2          <dbl> -0.0132859442, 0.0017698221, -0~
## $ feat_local_BLOSUM_scl3.1.lag2          <dbl> -0.0113681100, -0.0038454663, -~
## $ feat_local_BLOSUM_scl4.1.lag2          <dbl> -1.050142e-05, -1.696546e-02, 6~
## $ feat_local_BLOSUM_scl5.1.lag2          <dbl> -0.0044735824, 0.0020303587, -0~
## $ feat_local_BLOSUM_scl3.2.lag2          <dbl> 0.0172920476, 0.0034218725, -0.~
## $ feat_local_BLOSUM_scl4.2.lag2          <dbl> 1.619270e-03, 1.018492e-02, 4.1~
## $ feat_local_BLOSUM_scl5.2.lag2          <dbl> 0.0011399232, 0.0010228948, -0.~
## $ feat_local_BLOSUM_scl4.3.lag2          <dbl> 0.0028140262, -0.0171954192, 0.~
## $ feat_local_BLOSUM_scl5.3.lag2          <dbl> -0.0185580238, 0.0076250262, -0~
## $ feat_local_BLOSUM_scl5.4.lag2          <dbl> 9.167118e-03, -5.455275e-03, -2~
## $ feat_local_BLOSUM_scl1.2.lag3          <dbl> -0.0018127750, -0.0096626384, -~
## $ feat_local_BLOSUM_scl1.3.lag3          <dbl> 0.0192562512, -0.0113219921, 0.~
## $ feat_local_BLOSUM_scl1.4.lag3          <dbl> -0.008020423, -0.001210117, 0.0~
## $ feat_local_BLOSUM_scl1.5.lag3          <dbl> -0.009677449, 0.017012067, -0.0~
## $ feat_local_BLOSUM_scl2.3.lag3          <dbl> 1.406243e-02, -4.028121e-03, 2.~
## $ feat_local_BLOSUM_scl2.4.lag3          <dbl> -0.0003904314, 0.0102757328, 0.~
## $ feat_local_BLOSUM_scl2.5.lag3          <dbl> -0.0091141996, 0.0082001935, -0~
## $ feat_local_BLOSUM_scl3.4.lag3          <dbl> 0.0059146737, -0.0263977151, 0.~
## $ feat_local_BLOSUM_scl3.5.lag3          <dbl> -0.0206431704, 0.0118661880, -0~
## $ feat_local_BLOSUM_scl4.5.lag3          <dbl> 0.0193235865, -0.0117302746, -0~
## $ feat_local_BLOSUM_scl2.1.lag3          <dbl> -2.322579e-03, -1.350953e-02, 8~
## $ feat_local_BLOSUM_scl3.1.lag3          <dbl> 0.0129436159, 0.0276420582, -0.~
## $ feat_local_BLOSUM_scl4.1.lag3          <dbl> -0.0045148155, 0.0162318137, 0.~
## $ feat_local_BLOSUM_scl5.1.lag3          <dbl> 0.0046241738, -0.0046952823, 0.~
## $ feat_local_BLOSUM_scl3.2.lag3          <dbl> -0.0172597435, 0.0115726637, 0.~
## $ feat_local_BLOSUM_scl4.2.lag3          <dbl> -0.0078634075, 0.0189804062, -0~
## $ feat_local_BLOSUM_scl5.2.lag3          <dbl> 1.908895e-03, -1.768445e-03, 2.~
## $ feat_local_BLOSUM_scl4.3.lag3          <dbl> -1.966188e-02, 1.197465e-02, -1~
## $ feat_local_BLOSUM_scl5.3.lag3          <dbl> -0.023166367, -0.010435371, 0.0~
## $ feat_local_BLOSUM_scl5.4.lag3          <dbl> 0.0040344731, 0.0003782294, 0.0~
## $ feat_local_SOCN_Schneider.lag1         <dbl> 3.170780, 5.724153, 5.737633, 5~
## $ feat_local_SOCN_Schneider.lag2         <dbl> 4.132290, 6.048437, 4.347497, 3~
## $ feat_local_SOCN_Schneider.lag3         <dbl> 3.368122, 6.308152, 3.792324, 4~
## $ feat_local_SOCN_Grantham.lag1          <dbl> 124613.22, 120410.55, 218400.86~
## $ feat_local_SOCN_Grantham.lag2          <dbl> 140864.44, 140317.65, 168769.57~
## $ feat_local_SOCN_Grantham.lag3          <dbl> 94633.56, 106635.10, 169291.79,~
## $ feat_local_QSO_Schneider.Xr.A          <dbl> 1.14964060, 0.59648207, 0.48327~
## $ feat_local_QSO_Schneider.Xr.R          <dbl> 0.25421948, 0.94448484, 0.29303~
## $ feat_local_QSO_Schneider.Xr.N          <dbl> 0.87438397, 0.00000000, 0.36280~
## $ feat_local_QSO_Schneider.Xr.D          <dbl> 0.00000000, 0.00000000, 0.66172~
## $ feat_local_QSO_Schneider.Xr.C          <dbl> 0.00000000, 0.00000000, 0.39157~
## $ feat_local_QSO_Schneider.Xr.E          <dbl> 0.47908321, 0.00000000, 0.14732~
## $ feat_local_QSO_Schneider.Xr.Q          <dbl> 1.0235860, 0.1056440, 0.0000000~
## $ feat_local_QSO_Schneider.Xr.G          <dbl> 1.03736069, 0.58839781, 0.00000~
## $ feat_local_QSO_Schneider.Xr.H          <dbl> 0.00000000, 0.00000000, 0.00000~
## $ feat_local_QSO_Schneider.Xr.I          <dbl> 0.33226539, 0.25769334, 0.00000~
## $ feat_local_QSO_Schneider.Xr.L          <dbl> 0.48695628, 0.16124193, 0.26433~
## $ feat_local_QSO_Schneider.Xr.K          <dbl> 0.00000000, 0.55803248, 0.23836~
## $ feat_local_QSO_Schneider.Xr.M          <dbl> 0.15470430, 0.00000000, 0.00000~
## $ feat_local_QSO_Schneider.Xr.F          <dbl> 0.00000000, 0.05415185, 0.08692~
## $ feat_local_QSO_Schneider.Xr.P          <dbl> 0.43728282, 0.31905049, 0.72311~
## $ feat_local_QSO_Schneider.Xr.S          <dbl> 0.0000000, 0.9086149, 0.8086763~
## $ feat_local_QSO_Schneider.Xr.T          <dbl> 0.53978612, 0.26243758, 0.00000~
## $ feat_local_QSO_Schneider.Xr.W          <dbl> 0.48695628, 0.00000000, 0.00000~
## $ feat_local_QSO_Schneider.Xr.Y          <dbl> 0.0000000, 0.1531008, 1.3576115~
## $ feat_local_QSO_Schneider.Xr.V          <dbl> 0.04811902, 0.47158959, 0.47735~
## $ feat_local_QSO_Grantham.Xr.A           <dbl> 6.486480e-05, 4.809404e-05, 2.0~
## $ feat_local_QSO_Grantham.Xr.R           <dbl> 1.534710e-05, 7.242176e-05, 1.3~
## $ feat_local_QSO_Grantham.Xr.N           <dbl> 4.946007e-05, 0.000000e+00, 1.5~
## $ feat_local_QSO_Grantham.Xr.D           <dbl> 0.000000e+00, 0.000000e+00, 2.8~
## $ feat_local_QSO_Grantham.Xr.C           <dbl> 0.000000e+00, 0.000000e+00, 1.6~
## $ feat_local_QSO_Grantham.Xr.E           <dbl> 2.756312e-05, 0.000000e+00, 6.3~
## $ feat_local_QSO_Grantham.Xr.Q           <dbl> 5.875896e-05, 7.286597e-06, 0.0~
## $ feat_local_QSO_Grantham.Xr.G           <dbl> 5.876206e-05, 4.423150e-05, 0.0~
## $ feat_local_QSO_Grantham.Xr.H           <dbl> 0.000000e+00, 0.000000e+00, 0.0~
## $ feat_local_QSO_Grantham.Xr.I           <dbl> 1.855945e-05, 1.938233e-05, 0.0~
## $ feat_local_QSO_Grantham.Xr.L          <dbl> 2.778888e-05, 1.085858e-05, 1.2~
## $ feat_local_QSO_Grantham.Xr.K          <dbl> 0.000000e+00, 4.447639e-05, 9.9~
## $ feat_local_QSO_Grantham.Xr.M          <dbl> 9.280518e-06, 0.000000e+00, 0.0~
## $ feat_local_QSO_Grantham.Xr.F          <dbl> 0.000000e+00, 3.658148e-06, 3.8~
## $ feat_local_QSO_Grantham.Xr.P          <dbl> 2.460767e-05, 2.575255e-05, 3.1~
## $ feat_local_QSO_Grantham.Xr.S          <dbl> 0.000000e+00, 6.918182e-05, 3.4~
## $ feat_local_QSO_Grantham.Xr.T          <dbl> 3.105954e-05, 2.019108e-05, 0.0~
## $ feat_local_QSO_Grantham.Xr.W          <dbl> 2.778888e-05, 0.000000e+00, 0.0~
## $ feat_local_QSO_Grantham.Xr.Y          <dbl> 0.000000e+00, 1.113982e-05, 5.7~
## $ feat_local_QSO_Grantham.Xr.V          <dbl> 2.992094e-06, 3.750310e-05, 2.0~
## $ feat_local_QSO_Schneider.Xd.1         <dbl> 0.1528770, 0.2017357, 0.2402734~
## $ feat_local_QSO_Schneider.Xd.2         <dbl> 0.1983321, 0.2151919, 0.1816808~
## $ feat_local_QSO_Schneider.Xd.3         <dbl> 0.1618347, 0.2243443, 0.1583053~
## $ feat_local_QSO_Grantham.Xd.1          <dbl> 0.3460884, 0.3281892, 0.3939739~
## $ feat_local_QSO_Grantham.Xd.2          <dbl> 0.3911430, 0.3812736, 0.3049589~
## $ feat_local_QSO_Grantham.Xd.3          <dbl> 0.2627408, 0.2905096, 0.3010491~
## $ feat_local_ScalesGap_scl1.lag1        <dbl> -26.805487, -4.041661, -32.7946~
## $ feat_local_ScalesGap_scl2.lag1        <dbl> 7.1132015, 9.6892809, 4.9705467~
## $ feat_local_ScalesGap_scl3.lag1        <dbl> -14.3540200, -16.9134273, 4.146~
## $ feat_local_ScalesGap_scl4.lag1        <dbl> 2.541916, 6.467220, 2.382790, 1~
## $ feat_local_ScalesGap_scl5.lag1        <dbl> 0.2304271, 4.4648359, 0.5195277~
## $ feat_local_ScalesGap_scl1.lag2        <dbl> -24.061876, -7.508552, -10.9341~
## $ feat_local_ScalesGap_scl2.lag2        <dbl> 6.1390176, 9.0849711, 4.7028678~
## $ feat_local_ScalesGap_scl3.lag2        <dbl> 19.290653, -34.734158, 7.198661~
## $ feat_local_ScalesGap_scl4.lag2        <dbl> -24.5640352, -1.9927625, 20.545~
## $ feat_local_ScalesGap_scl5.lag2        <dbl> -3.2155963, 1.4643845, -11.3028~
## $ feat_local_ScalesGap_scl1.lag3        <dbl> 29.030296174, -16.393519055, 9.~
## $ feat_local_ScalesGap_scl2.lag3        <dbl> 7.8989149, 5.8110418, 10.351730~
## $ feat_local_ScalesGap_scl3.lag3        <dbl> -1.4050557, -0.8491584, 10.5552~
## $ feat_local_ScalesGap_scl4.lag3        <dbl> 0.1653006, 6.3774369, -5.307702~
## $ feat_local_ScalesGap_scl5.lag3        <dbl> -9.9989795, -0.8421555, -4.8963~
## $ feat_local_ScalesGap_scl1.2.lag1      <dbl> 0.3493892, 3.1733162, 5.8309819~
## $ feat_local_ScalesGap_scl1.3.lag1      <dbl> 7.097885, -30.900077, 11.276234~
## $ feat_local_ScalesGap_scl1.4.lag1      <dbl> -45.6704719, -12.9432018, -1.72~
## $ feat_local_ScalesGap_scl1.5.lag1      <dbl> -12.997965, -1.757009, -31.9418~
## $ feat_local_ScalesGap_scl2.3.lag1      <dbl> 2.7963360, 14.1012512, 2.306771~
## $ feat_local_ScalesGap_scl2.4.lag1      <dbl> 9.9997841, 7.5760184, 6.6040670~
## $ feat_local_ScalesGap_scl2.5.lag1      <dbl> 4.4652100, -0.9386516, 6.030180~
## $ feat_local_ScalesGap_scl3.4.lag1      <dbl> 36.558938, 11.607218, -21.98668~
## $ feat_local_ScalesGap_scl3.5.lag1      <dbl> -5.8252074, 5.5950041, -2.59134~
## $ feat_local_ScalesGap_scl4.5.lag1      <dbl> 18.4609973, 0.6535935, 4.034194~
## $ feat_local_ScalesGap_scl2.1.lag1      <dbl> -2.3709568, -14.2648145, 1.4983~
## $ feat_local_ScalesGap_scl3.1.lag1      <dbl> 17.2987064, 27.7208033, -3.1113~
## $ feat_local_ScalesGap_scl4.1.lag1      <dbl> 13.718060, 8.590215, 17.471048,~
## $ feat_local_ScalesGap_scl5.1.lag1      <dbl> 1.6667712, 1.9797268, 5.4449267~
## $ feat_local_ScalesGap_scl3.2.lag1      <dbl> 1.8843181, 1.7271123, 8.4086830~
## $ feat_local_ScalesGap_scl4.2.lag1      <dbl> 2.2110154, 4.1175677, -4.638707~
## $ feat_local_ScalesGap_scl5.2.lag1      <dbl> 0.8582824, -0.5743568, -6.33126~
## $ feat_local_ScalesGap_scl4.3.lag1      <dbl> -4.9472747, 1.6081799, -16.3283~
## $ feat_local_ScalesGap_scl5.3.lag1      <dbl> -5.86430209, 6.59444033, 4.3693~
## $ feat_local_ScalesGap_scl5.4.lag1      <dbl> 0.2410665, -9.3777793, 5.736627~
## $ feat_local_ScalesGap_scl1.2.lag2      <dbl> 4.4404459, -4.0732796, 2.165522~
## $ feat_local_ScalesGap_scl1.3.lag2      <dbl> -20.6278097, -9.9278451, -6.443~
## $ feat_local_ScalesGap_scl1.4.lag2       <dbl> 1.0564451, -9.1868634, 12.03543~
## $ feat_local_ScalesGap_scl1.5.lag2       <dbl> -6.88679333, 5.88113394, 4.7210~
## $ feat_local_ScalesGap_scl2.3.lag2       <dbl> 8.93930748, 17.40772214, 1.8680~
## $ feat_local_ScalesGap_scl2.4.lag2       <dbl> 8.14665923, 8.61293260, -2.4148~
## $ feat_local_ScalesGap_scl2.5.lag2       <dbl> 1.9054839, -2.7492362, 4.404310~
## $ feat_local_ScalesGap_scl3.4.lag2       <dbl> -21.0326845, 0.8996968, -9.3752~
## $ feat_local_ScalesGap_scl3.5.lag2       <dbl> 10.9355775, 3.7855127, -0.25939~
## $ feat_local_ScalesGap_scl4.5.lag2       <dbl> -13.4261067, -7.1171604, 5.1068~
## $ feat_local_ScalesGap_scl2.1.lag2       <dbl> -2.6124475, -13.4240392, 6.3043~
## $ feat_local_ScalesGap_scl3.1.lag2       <dbl> -3.44656032, 21.12687281, -14.3~
## $ feat_local_ScalesGap_scl4.1.lag2       <dbl> -18.7485642, -15.4262755, -0.19~
## $ feat_local_ScalesGap_scl5.1.lag2       <dbl> 20.0953017, -9.7116574, 7.62952~
## $ feat_local_ScalesGap_scl3.2.lag2       <dbl> -0.5438178, 8.6345267, 6.490684~
## $ feat_local_ScalesGap_scl4.2.lag2       <dbl> 10.983856, 14.417217, -4.055239~
## $ feat_local_ScalesGap_scl5.2.lag2       <dbl> -2.1341124, 7.2562862, -7.82862~
## $ feat_local_ScalesGap_scl4.3.lag2       <dbl> 2.9322826, -13.1131824, -6.3415~
## $ feat_local_ScalesGap_scl5.3.lag2       <dbl> -10.92025754, -9.19500577, 9.91~
## $ feat_local_ScalesGap_scl5.4.lag2       <dbl> -6.205576, -1.473374, -2.128489~
## $ feat_local_ScalesGap_scl1.2.lag3       <dbl> -7.641048, -9.079270, -5.115167~
## $ feat_local_ScalesGap_scl1.3.lag3       <dbl> -6.792791, 21.711626, -16.60704~
## $ feat_local_ScalesGap_scl1.4.lag3       <dbl> 7.1035918, -7.3868414, -15.8980~
## $ feat_local_ScalesGap_scl1.5.lag3       <dbl> 12.6776225, -9.5224158, 8.38404~
## $ feat_local_ScalesGap_scl2.3.lag3       <dbl> 5.007996, 1.428866, 9.445487, -~
## $ feat_local_ScalesGap_scl2.4.lag3       <dbl> 2.72305578, 16.45374617, 9.3740~
## $ feat_local_ScalesGap_scl2.5.lag3       <dbl> 3.06325892, 1.61799129, -3.9728~
## $ feat_local_ScalesGap_scl3.4.lag3       <dbl> 6.4467686, -19.0477989, -7.5843~
## $ feat_local_ScalesGap_scl3.5.lag3       <dbl> -30.15346358, 8.91789551, -2.75~
## $ feat_local_ScalesGap_scl4.5.lag3       <dbl> -6.5920550, -4.0006191, 14.8710~
## $ feat_local_ScalesGap_scl2.1.lag3       <dbl> -13.1478338, 13.4166676, -17.93~
## $ feat_local_ScalesGap_scl3.1.lag3       <dbl> -13.670038, -71.389934, 26.4687~
## $ feat_local_ScalesGap_scl4.1.lag3       <dbl> -10.9217951, -27.0934227, -9.53~
## $ feat_local_ScalesGap_scl5.1.lag3       <dbl> -9.5055855, -3.8591734, -14.353~
## $ feat_local_ScalesGap_scl3.2.lag3       <dbl> 9.917971, 23.934860, -3.646627,~
## $ feat_local_ScalesGap_scl4.2.lag3       <dbl> 10.9276507, 12.8712718, 5.79240~
## $ feat_local_ScalesGap_scl5.2.lag3       <dbl> 2.57769040, 4.80258361, 2.22407~
## $ feat_local_ScalesGap_scl4.3.lag3       <dbl> -5.571386, 3.015052, -22.406411~
## $ feat_local_ScalesGap_scl5.3.lag3       <dbl> 1.7394191, -6.2758328, 7.380258~
## $ feat_local_ScalesGap_scl5.4.lag3       <dbl> -12.73121547, -4.85432444, -33.~
## $ Class                                  <dbl> 1, 1, 1, -1, -1, 1, -1, -1, 1, ~

# Categorizing columns based on prefixes in Task 2 Context
prefix_counts <- tibble(name = names(df)) %>%
  mutate(prefix = case_when(
    str_starts(name, "Info_") ~ "Info_",
    str_starts(name, "feat_") ~ "feat_",
    name == "Class" ~ "Class",
    TRUE ~ "other"
  )) %>%
  count(prefix, sort = TRUE)
prefix_counts
## # A tibble: 3 x 2
##   prefix     n
##   <chr>  <int>
## 1 feat_    385
## 2 Info_      4
## 3 Class      1
```

The dataset consists of 7403 observations and 390 variables. Of these, 385 are numerical feature columns (prefix 'feat_'), 4 information columns (prefix 'Info_') and one target variable 'Class'.

```
# EDA Step 2: Checking class distribution and labels
class_tab <- df %>%
  count(Class) %>%
  arrange(desc(n)) %>%
  mutate(prop = n / sum(n))
print(class_tab)
## # A tibble: 2 x 3
##   Class     n  prop
##   <dbl> <int> <dbl>
## 1    -1  4395 0.594
## 2     1  3008 0.406

# Converting Class to a factor
df <- df %>%
  mutate(Class = factor(Class))
```

The target variable Class is moderately balanced with approximately 59% of observations in class '-1' and 41% in class '1'. It has been converted to a factor to ensure proper model handling.

```
# EDA Step 3: Checking for missing values

# Total number of missing values in the dataset
total_missing <- sum(is.na(df))
total_missing
## [1] 486

# Count missing values for each variable
missing_summary <- df %>%
  summarise(across(everything(), ~ sum(is.na(.)))) %>%
  pivot_longer(everything(), names_to = "variable", values_to = "missing") %>%
  filter(missing > 0) %>%
  arrange(desc(missing))

missing_summary
## # A tibble: 267 x 2
##    variable                                missing
##    <chr>                                     <int>
##  1 feat_local_CTDC_secondarystruct.Group3        5
##  2 feat_local_CTDD_prop3.G1.residue25            5
##  3 feat_local_CTDD_prop4.G1.residue100           5
##  4 feat_local_ScalesGap_scl1.3.lag1              5
##  5 feat_local_Entropy                            4
##  6 feat_local_CTDC_polarizability.Group3         4
##  7 feat_local_CTDD_prop1.G2.residue75            4
##  8 feat_local_CTDD_prop3.G2.residue75            4
##  9 feat_local_CTDD_prop5.G1.residue0             4
## 10 feat_local_CTDD_prop7.G3.residue75            4
## # i 257 more rows
```

```
# Identifying if there are any missing values in Class or any of the Info_ variables
"Class" %in% missing_summary$variable
## [1] FALSE

any(str_starts(missing_summary$variable, "Info_"))
## [1] TRUE

missing_summary %>%
  filter(str_starts(variable, "Info_"))
## # A tibble: 1 x 2
##   variable missing
##   <chr>      <int>
## 1 Info_pos       1
```

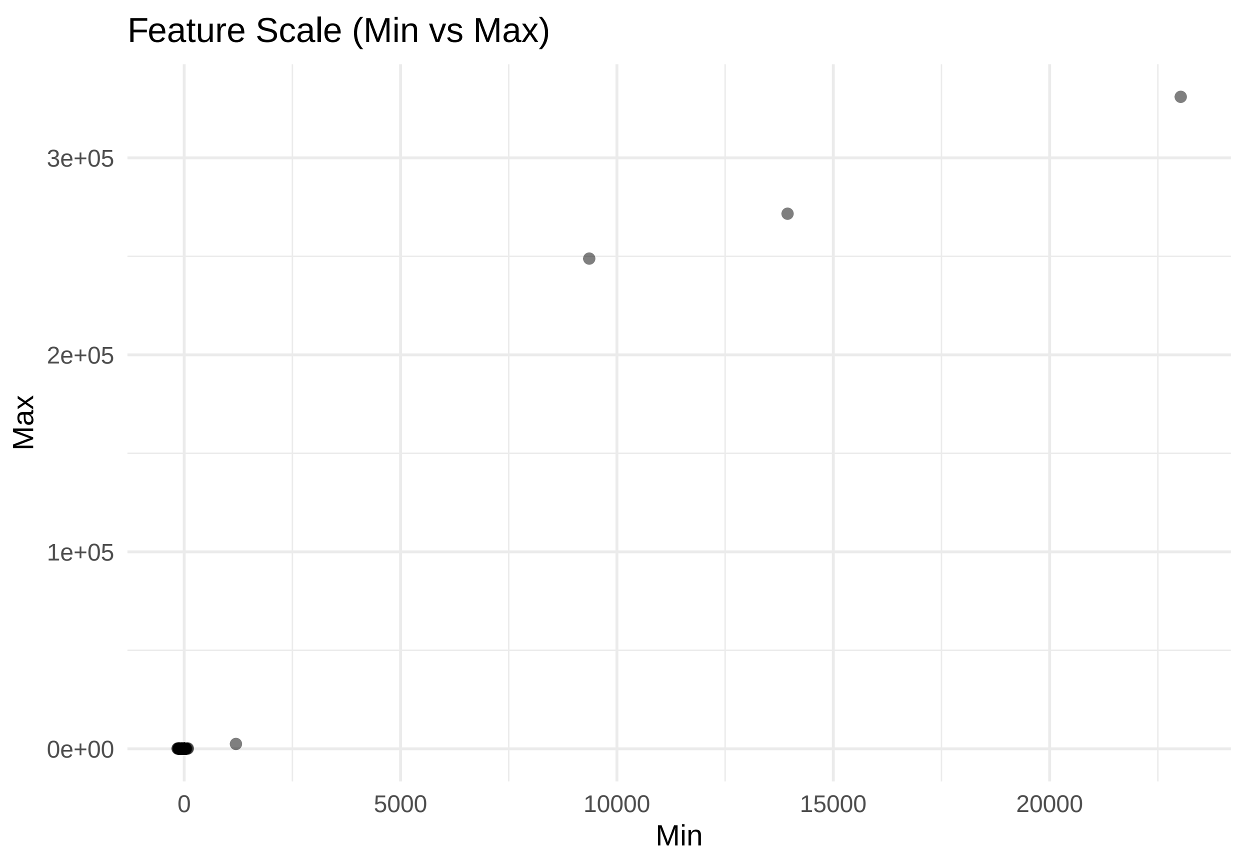
A total of 486 missing values were found across the dataset. Missingness is sparse because this is much smaller than the total number of cells in the dataset and no individual variable contains more than 5 missing values. We have also verified that there are no missing values in the target variable 'Class' and only one 'Info_' column contains a missing value. Given the low and unstructured nature of the missing values, imputation will be sufficient during preprocessing.

```
# EDA Step 4: Scale, variance, outliers and correlation of numeric features

# Summarizing min and max values for each numerical feature
df_num <- df %>% select(starts_with("feat_"))

scale_summary <- df_num %>%
  summarise(across(everything(),
                  list(min = ~min(.x, na.rm = TRUE),
                       max = ~max(.x, na.rm = TRUE)))) %>%
  pivot_longer(everything(),
               names_to = c("variable", "stat"),
               names_pattern = "(.*)_(min|max)$") %>%
  pivot_wider(names_from = stat, values_from = value)

scale_summary %>%
  ggplot(aes(x = min, y = max)) +
  geom_point(alpha = 0.5) +
  theme_minimal() +
  labs(title = "Feature Scale (Min vs Max)",
       x = "Min",
       y = "Max")
```



```
# Determining global min and max values across all numerical features
all_vals   <- unlist(df_num)
global_min <- min(all_vals, na.rm = TRUE)
global_max <- max(all_vals, na.rm = TRUE)

cat("Global numeric feature range: [",global_min, ", ",global_max,"]\n")
## Global numeric feature range: [ -152.3783 ,  330982.6 ]

# Checking for variables with near zero variance
nzv_check <- df_num %>%
  summarise(across(everything(), ~ length(unique(.)))) %>%
  pivot_longer(cols = everything(),
               names_to = "feature",
               values_to = "n_unique") %>%
  arrange(n_unique)

head(nzv_check)
## # A tibble: 6 x 2
##   feature                           n_unique
##   <chr>                                <int>
## 1 feat_local_AAC_W                       744
## 2 feat_local_CTDT_prop5.Tr1331           756
## 3 feat_local_CTDD_prop5.G2.residue0      776
## 4 feat_local_CTDD_prop5.G2.residue100    788
## 5 feat_local_AAC_H                       849
## 6 feat_local_AAC_M                       854
```

```
# Checking for extreme values per feature for outliers using z-score
outlier_check <- df_num %>%
  summarise(across(
    everything(),
    ~ mean(abs(scale(.x)) > 3, na.rm = TRUE)
  )) %>%
  pivot_longer(everything(),
               names_to = "feature",
               values_to = "prop_outliers") %>%
  arrange(desc(prop_outliers))

summary(outlier_check$prop_outliers)
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.000000 0.007296 0.010536 0.011039 0.014456 0.025000

# Checking for correlation between numerical features
corr_mat <- cor(df_num, use = "pairwise.complete.obs")
corr_df <- as.data.frame(as.table(corr_mat)) %>% setNames(c("Var1","Var2","Corr")) %>%
  filter(Var1 != Var2) %>% mutate(AbsCorr = abs(Corr)) %>% arrange(desc(AbsCorr))

cat("Top correlated pairs (abs corr):\n")
## Top correlated pairs (abs corr):
print(head(corr_df, 10))
##                                      Var1
## 1                 feat_local_AAtypes_Polar
## 2              feat_local_AAtypes_NonPolar
## 3            feat_local_CTDC_charge.Group3
## 4     feat_local_CTDC_solventaccess.Group2
## 5    feat_local_CTDC_polarizability.Group3
## 6   feat_local_CTDC_normwaalsvolume.Group3
## 7                feat_local_AAtypes_Acidic
## 8    feat_local_CTDC_hydrophobicity.Group1
## 9        feat_local_CTDD_prop7.G2.residue0
## 10      feat_local_CTDD_prop7.G2.residue25
##                                      Var2 Corr AbsCorr
## 1              feat_local_AAtypes_NonPolar   -1       1
## 2                 feat_local_AAtypes_Polar   -1       1
## 3                feat_local_AAtypes_Acidic    1       1
## 4    feat_local_CTDC_hydrophobicity.Group1    1       1
## 5   feat_local_CTDC_normwaalsvolume.Group3    1       1
## 6    feat_local_CTDC_polarizability.Group3    1       1
## 7            feat_local_CTDC_charge.Group3    1       1
## 8     feat_local_CTDC_solventaccess.Group2    1       1
## 9        feat_local_CTDD_prop1.G1.residue0    1       1
## 10      feat_local_CTDD_prop1.G1.residue25    1       1
```

We can see that the numerical features vary widely in scale. Most features have min and max values clustered near zero but some take very large positive values and some take negative values. Due to the high dimensionality of the dataset, we have used a min-max scatter plot to check the scale of values. Over all values the global range is approximately [-152, 3.3e5]. These extreme values suggest that scaling will be required. From our near zero variance check, we can see that all features exhibit substantial variability, therefore we do not need to filter out any of these features in further steps.

To detect outliers we computed the proportion of observations with Z score magnitude greater than 3 for each numerical feature. Most features have between 0% and 2.5% such values which indicates some extreme observations but only in small proportions. The correlation check shows several pairs of highly correlated features which will be addressed by PCA during modelling.

Overall, the EDA shows that the dataset is high-dimensional with a moderately balanced target variable. Missing values are sparse, so simple imputation will be sufficient. The numerical features exhibit substantial difference in scale, so scaling will be required before modelling. All features show adequate variability, so we do not need any feature filtering. The proportion of extreme values is small, so outlier removal is unnecessary.

---

2. Data preprocessing and feature engineering

Note: It is expected that this section should result in no more than 400-ish words (possibly less; not counting the code blocks).

Add here your data pre-processing and feature selection/engineering steps. Your solution should include both your code (which markers must be able to run independently) and your rationale. In general, your steps should include:

- A short justification of which insight from your EDA is motivating a given data transformation.
- A code block implementing your pre-processing step.

In addition to the regular data transformations emerging from the insights you gathered during EDA, this section must contain the splitting off of a test set that you'll use later for final performance assessment (check the CW specs for details).

At the end of this section you should also include a short commentary on what changes the pre-processing has induced in your data (i.e., what did the data "look like" before this step vs. what it "looks like" afterwards.

You can use as many code blocks as needed, but don't overdo it (remember - everything needs to have a proper rationale). If you compare different pre-processing options, make sure to make it clear, and to indicate at the end which option you selected, and why.

Note: In this section you'll be basically defining which data transformations you will later incorporate into your predictive pipeline. When you build the pipeline in the next section, you'll encapsulate all those steps (together with the model) into a single object that you can fit and deploy as a single unit.

We first split the data into train and test splits by group on Info_group to prevent data leakage.

```r
# DPP Step 1: Splitting data into train and test sets by group
set.seed(MY_STUDENT_ID)

# 80:20 split into training and test based on Info_group
data_split <- group_initial_split(df, group = Info_group, prop = 0.8)

df_train <- training(data_split)
df_test <- testing(data_split)

length(intersect(df_train$Info_group, df_test$Info_group))
## [1] 0
```

In the construction of our recipe we note from EDA: 1. Info_ columns are non informative as predictors and should be excluded from the predictor set. 2. Missing values are sparse and unstructured, so trimmed mean imputation is sufficient. 3. Numerical features have very different scales so we apply normalization. 4. Small proportion of outliers across features tells us that Winsorisation is not necessary.

```r
# DPP Step 2: Preprocessing recipe based on EDA
```

```r
pp_recipe <- recipe(Class ~ ., data = df_train) %>%
  step_rm(starts_with("Info_")) %>%
  step_impute_mean(all_numeric_predictors(), trim = 0.1) %>%
  step_normalize(all_numeric_predictors())
pp_recipe
##
## -- Recipe ----------------------------------------------------------------------
##
## -- Inputs
## Number of variables by role
## outcome:     1
## predictor: 389
##
## -- Operations
## * Variables removed: starts_with("Info_")
## * Mean imputation for: all_numeric_predictors()
## * Centering and scaling for: all_numeric_predictors()

pp_recipe_prepped <- prep(pp_recipe)
pp_recipe_prepped
##
## -- Recipe ----------------------------------------------------------------------
##
## -- Inputs
## Number of variables by role
## outcome:     1
## predictor: 389
##
## -- Training information
## Training data contained 5897 data points and 379 incomplete rows.
##
## -- Operations
## * Variables removed: Info_PepID, Info_group, Info_protein_id, ... | Trained
## * Mean imputation for: feat_local_Entropy, ... | Trained
## * Centering and scaling for: feat_local_Entropy, ... | Trained

train_processed <- bake(pp_recipe_prepped, df_train)
train_processed
## # A tibble: 5,897 x 386
##    feat_local_Entropy feat_local_MolWeight feat_local_AAtypes_Tiny
##                 <dbl>                <dbl>                   <dbl>
##  1              0.138              -0.594                    0.521
##  2              0.242               0.967                   -0.401
##  3             -1.01                0.846                   -0.124
##  4              0.913               0.197                   -0.138
##  5              0.830              -0.437                    0.337
##  6              0.534              -0.490                    0.405
##  7              0.113               0.0962                   0.644
##  8             -0.291              -0.352                    1.15
##  9             -0.288              -0.624                   -1.19
## 10              0.535              -0.0270                  -0.682
## # i 5,887 more rows
## # i 383 more variables: feat_local_AAtypes_Small <dbl>,
## #   feat_local_AAtypes_Aliphatic <dbl>, feat_local_AAtypes_Aromatic <dbl>,
## #   feat_local_AAtypes_NonPolar <dbl>, feat_local_AAtypes_Polar <dbl>,
## #   feat_local_AAtypes_Charged <dbl>, feat_local_AAtypes_Basic <dbl>,
## #   feat_local_AAtypes_Acidic <dbl>, feat_local_Atoms_nC <dbl>,
## #   feat_local_Atoms_nH <dbl>, feat_local_Atoms_nN <dbl>, ...

cat("Info_ columns ",
    any(startsWith(names(train_processed), "Info_")), "\n")
## Info_ columns  FALSE
cat("Total remaining NAs after preprocessing:",
    sum(is.na(train_processed)), "\n")
## Total remaining NAs after preprocessing: 0
cat("Number of predictors after preprocessing (excluding Class):",
    ncol(train_processed) - 1, "\n")
## Number of predictors after preprocessing (excluding Class): 385
```

The preprocessing recipe removes all non-informative Info_ columns from the predictor set, imputes missing values using winsorised mean (10% trim) and normalizes all numeric features. The only unaddressed insight from EDA is the presence of highly correlated features, which we will address during modelling using PCA.

---

3. Modelling

Note: It is expected that this section should result in no more than 400-ish words (possibly less; not counting the code blocks).

Add here your modelling steps. Your solution should include both your code (which markers must be able to run independently) and your rationale. This section must include:

- The fitting of a preliminary model on your preprocessed training data (to estimate some baseline performance level and to make sure your data is ready for modelling). Make this a simple model - logistic regression, decision trees or kNN are good options (but other may be used instead, if you prefer).
- The building of a predictive pipeline encapsulating all the learned/learnable aspects of your solution. This should result in a pipeline-type object that uses your raw training data (post EDA and test set splitting, but prior to any preprocessing), fits all your preprocessing transformations (incl. dimensionality reduction) and the predictive model using that data, and can then be used to ingest your test data (or any new data in the same format) and generate predictions for each entry.

Please refer to the CW specs for other requirements of this step (including, e.g., the addition of a dimensionality reduction step into your pipeline).

You can use as many code blocks as needed, but don't overdo it (remember - everything needs to have a proper rationale). If you compare different models, make sure to make it clear, and to indicate at the end which pipeline option is your final selected one.

```
# Modelling Step 1: Simple logistic regression using DPP recipe
lr_model <- logistic_reg() %>%
  set_mode("classification") %>%
  set_engine("glm")

lr_wf <- workflow() %>%
  add_recipe(pp_recipe) %>%
  add_model(lr_model)

lr_fit <- lr_wf %>% fit(df_train)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
lr_fit
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: logistic_reg()
##
## -- Preprocessor ----------------------------------------------------------------
## 3 Recipe Steps
##
## * step_rm()
## * step_impute_mean()
## * step_normalize()
##
## -- Model -----------------------------------------------------------------------
##
## Call:  stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)
##
## Coefficients:
##                      (Intercept)                 feat_local_Entropy
##                       -7.704e-01                          6.919e-02
##              feat_local_MolWeight             feat_local_AAtypes_Tiny
##                        1.374e+01                         -1.561e+01
##          feat_local_AAtypes_Small        feat_local_AAtypes_Aliphatic
##                        3.059e+02                         -1.223e+00
##       feat_local_AAtypes_Aromatic         feat_local_AAtypes_NonPolar
##                        3.824e+02                         -1.858e+01
##          feat_local_AAtypes_Polar          feat_local_AAtypes_Charged
##                       -7.050e-01                         -3.484e+03
##          feat_local_AAtypes_Basic           feat_local_AAtypes_Acidic
##                        1.327e+02                          1.365e+01
##               feat_local_Atoms_nC                 feat_local_Atoms_nH
##                       -1.778e+01                          2.414e+00
##               feat_local_Atoms_nN                 feat_local_Atoms_nO
##                       -2.134e+00                         -1.592e+00
##               feat_local_Atoms_nS                    feat_local_AAC_A
##                       -2.301e+01                         -3.071e+01
##                  feat_local_AAC_R                    feat_local_AAC_N
##                        5.977e-01                         -1.450e+03
##                  feat_local_AAC_D                    feat_local_AAC_C
##                        8.628e+01                         -1.181e+00
##                  feat_local_AAC_E                    feat_local_AAC_Q
##                               NA                         -1.396e+03
##                  feat_local_AAC_G                    feat_local_AAC_H
##                        2.437e+02                          9.257e+02
##                  feat_local_AAC_I                    feat_local_AAC_L
##                        1.382e+02                         -2.308e+01
##                  feat_local_AAC_K                    feat_local_AAC_M
##                        1.695e+00                         -3.563e+01
##                  feat_local_AAC_F                    feat_local_AAC_P
##                        5.235e+00                         -5.096e+01
##                  feat_local_AAC_S                    feat_local_AAC_T
##                        6.225e+00                         -4.703e+01
##                  feat_local_AAC_W                    feat_local_AAC_Y
##                              4.722e+00                                1.989e+01
##                         feat_local_AAC_V   feat_local_CTDC_hydrophobicity.Group1
##                              2.520e+01                                6.064e-01
##  feat_local_CTDC_hydrophobicity.Group2   feat_local_CTDC_hydrophobicity.Group3
##                             -4.564e+03                               -3.732e+03
## feat_local_CTDC_normwaalsvolume.Group1  feat_local_CTDC_normwaalsvolume.Group2
##                             -4.224e+01                               -1.484e+02
## feat_local_CTDC_normwaalsvolume.Group3        feat_local_CTDC_polarity.Group1
##                             -2.706e+02                                9.073e+01
##       feat_local_CTDC_polarity.Group2        feat_local_CTDC_polarity.Group3
##                              2.993e+02                                       NA
##
## ...
## and 344 more lines.
```

We have fit a preliminary predictive model based on logistic regression to ensure that our data is ready for modelling. From this model, we see warnings of fitted probabilities, NA coefficients and extremely large weights. These are indicative of severe multicollinearity. This aligns with our EDA and confirms that dimensionality reduction via PCA is required.

```
# Modelling Step 2: PCA dimensionality reduction.
pca_recipe <- pp_recipe %>%
  step_pca(all_numeric_predictors(), num_comp = tune())
pca_recipe
##
## -- Recipe ----------------------------------------------------------------------
##
## -- Inputs
## Number of variables by role
## outcome:     1
## predictor: 389
##
## -- Operations
## * Variables removed: starts_with("Info_")
## * Mean imputation for: all_numeric_predictors()
## * Centering and scaling for: all_numeric_predictors()
## * PCA extraction with: all_numeric_predictors()
```

From our EDA and preliminary model, we learned that PCA is required and has been added to the recipe. PCA will transform our 385 correlated features into a smaller set of uncorrelated principal components. We have set this number of components to be tuned as a hyperparameter.

```
# Modelling Step 3: Grouped cross-validation to prevent data leakage
set.seed(MY_STUDENT_ID)

group_folds <- group_vfold_cv(
  df_train,
  group = Info_group,
  v = 5
)

group_folds
## # Group 5-fold cross-validation
## # A tibble: 5 x 2
##   splits              id
##   <list>             <chr>
## 1 <split [3729/2168]> Resample1
## 2 <split [5196/701]>  Resample2
## 3 <split [5485/412]>  Resample3
## 4 <split [4617/1280]> Resample4
## 5 <split [4561/1336]> Resample5
```

To prevent data leakage we have used a 5-fold cross-validation, where folds are constructed using 'Info_group'.

We have selected two candidate sophisticated models to test and decide on the better model. The selected models are Random Forest and Elastic Net Logistic Regression. They were selected due to their ability to work on high dimensional data.

```
# Modelling Step 4: Comparing Random Forest and Elastic Net

# Simple PCA recipe without tuning
# Setting num_comp = 30 to retain ~10% of numerical features and ensure RF's mtry=15 is tested fairly
simple_pca_recipe <- pp_recipe %>%
  step_pca(all_numeric_predictors(), num_comp = 30)

# 4a Simple Random Forest
rf_model_simple <- rand_forest(
  mtry  = 15,
  min_n = 5,
  trees = 500
) %>%
  set_mode("classification") %>%
  set_engine("ranger")

rf_wf_simple <- workflow() %>%
  add_recipe(simple_pca_recipe) %>%
  add_model(rf_model_simple)

set.seed(MY_STUDENT_ID)
rf_cv <- fit_resamples(
  rf_wf_simple,
  resamples = group_folds,
  metrics = metric_set(f_meas)
)

rf_results <- collect_metrics(rf_cv)
rf_results
## # A tibble: 1 x 6
##   .metric .estimator  mean     n std_err .config
##   <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 f_meas  binary     0.767     5  0.0200 pre0_mod0_post0

# 4b Simple Elastic Net
enet_model_simple <- logistic_reg(
  penalty = 0.01,
  mixture = 0.5
) %>%
  set_mode("classification") %>%
  set_engine("glmnet")
```

```
enet_wf_simple <- workflow() %>%
  add_recipe(simple_pca_recipe) %>%
  add_model(enet_model_simple)

set.seed(MY_STUDENT_ID)
enet_cv <- fit_resamples(
  enet_wf_simple,
  resamples = group_folds,
  metrics = metric_set(f_meas)
)

enet_results <- collect_metrics(enet_cv)
enet_results
## # A tibble: 1 x 6
##   .metric .estimator  mean     n std_err .config
##   <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 f_meas  binary     0.739     5  0.0167 pre0_mod0_post0
```

On comparing Random Forest and Elastic Net, we see that Random Forest has an F1 score of approximately 0.767 while Elastic Net has an F1 score of approximately 0.739. This tells us that Random Forest performs better than Elastic Net and is selected as our model for next steps.

```
# Modelling Step 5: Hyperparameter tuning for Random Forest Model

rf_model <- rand_forest(
  mtry = tune(),
  min_n = tune(),
  trees = 500
) %>%
  set_mode("classification") %>%
  set_engine("ranger")

rf_wf <- workflow() %>%
  add_recipe(pca_recipe) %>%
  add_model(rf_model)
rf_wf
## == Workflow ===================================================================
## Preprocessor: Recipe
## Model: rand_forest()
##
## -- Preprocessor ---------------------------------------------------------------
## 4 Recipe Steps
##
## * step_rm()
## * step_impute_mean()
## * step_normalize()
## * step_pca()
##
## -- Model ----------------------------------------------------------------------
## Random Forest Model Specification (classification)
##
## Main Arguments:
##   mtry = tune()
##   trees = 500
##   min_n = tune()
##
## Computational engine: ranger

rf_grid <- grid_regular(
  mtry(range = c(5L, 20L)),
  min_n(range = c(2L, 10L)),
  num_comp(range = c(20L, 60L)),
  levels = 2
)

set.seed(MY_STUDENT_ID)
rf_tuned <- tune_grid(
  rf_wf,
  resamples = group_folds,
  grid = rf_grid,
  metrics = metric_set(f_meas)
)

show_best(rf_tuned, metric = "f_meas", n = 5)
## # A tibble: 5 x 9
##    mtry min_n num_comp .metric .estimator  mean     n std_err .config
##   <int> <int>    <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1     5     2       60 f_meas  binary     0.781     5  0.0196 pre2_mod1_post0
## 2     5    10       60 f_meas  binary     0.781     5  0.0205 pre2_mod2_post0
## 3    20    10       60 f_meas  binary     0.774     5  0.0188 pre2_mod4_post0
## 4    20     2       60 f_meas  binary     0.774     5  0.0208 pre2_mod3_post0
## 5     5     2       20 f_meas  binary     0.761     5  0.0199 pre1_mod1_post0
```

We have defined a workflow that involves tuning 3 hyperparameters: mtry, min_n and num_comp. We also ensure we use grouped CV folds to prevent data leakage. The best configuration achieved an F1 score of 0.781 (std_error = 0.0196). Our hyperparameter tuning tells us that this configuration has: mtry = 5, min_n = 2 and num_comp = 60.

We then finalize the workflow with these values and refit on the full training set to obtain our final model.

```
# Modelling Step 6: Finalize the tuned Random Forest workflow
best_rf <- select_best(rf_tuned, metric = "f_meas")

final_rf_wf <- finalize_workflow(
  rf_wf,
  best_rf
)

# Fitting on full training data
final_rf_fit <- final_rf_wf %>% fit(df_train)
final_rf_fit
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: rand_forest()
##
## -- Preprocessor ----------------------------------------------------------------
## 4 Recipe Steps
##
## * step_rm()
## * step_impute_mean()
## * step_normalize()
## * step_pca()
##
## -- Model -----------------------------------------------------------------------
## Ranger result
##
## Call:
##  ranger::ranger(x = maybe_data_frame(x), y = y, mtry = min_cols(~5L,      x), num.trees = ~500, min.
##
## Type:                             Probability estimation
## Number of trees:                  500
## Sample size:                      5897
## Number of independent variables:  60
## Mtry:                             5
## Target node size:                 2
## Variable importance mode:         none
## Splitrule:                        gini
## OOB prediction error (Brier s.):  0.1856752
```

---

4. Model assessment

Note: It is expected that this section should result in no more than 100-ish words (possibly less; not counting the code blocks).

This is a short section in which you must estimate the generalisation performance of your model on the test set that you isolated at the start of your Data preprocessing step. It is also a chance for you to ensure that your pipeline object can take in new data and return the required predictions.

Use your pipeline to generate predictions for your test set, and calculate the observed performance using the F_1-score as your main performance indicator. You can also additionally report other performance metrics if you think it's relevant.

You can use as many code blocks as needed, but it's likely that this section can be completed with a single one. Make sure that your performance indicators are clearly echoed, so that a reader can easily check if after re-running your script. Add a short comment on the observed performance (is it good? Is it poor? Remember that you're not being assessed on the performance of your models, but on the soundness of your approach).

```r
# Model Assessment: Testing pipeline on test set
test_preds <- predict(final_rf_fit, new_data = df_test)
test_probs <- predict(final_rf_fit, new_data = df_test, type = "prob")

test_results <- df_test %>%
  select(Class) %>%
  bind_cols(test_preds, test_probs)

test_metrics <- test_results %>%
  metric_set(f_meas, accuracy)(truth = Class, estimate = .pred_class)
test_metrics
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 f_meas   binary         0.759
## 2 accuracy binary         0.697
```

The test metrics tell us that our model achieves an F1 score of 0.759 on the test set, close to the cross-validated estimate of ~0.78. This tells us that the model generalizes well.

---

5. Model deployment

Note: This section is not expected to have much text in it, only the code and the results.

Finally, in this section you'll generate predictions for some data for which you do not know the class labels. This simulates the real-life scenario in which your model needs to be deployed to generate new predictions for new, unknown data (after all, if the labels were known, we wouldn't need a model).

Make sure that the data file **df_holdout.rds** is placed in the same folder as this Task02.Rmd file. The new data will then be automatically loaded and set up for you, and stored into a dataframe variable called `df_holdout`, with the same columns as the original `df` that was loaded at the start of this task.

- Use your fitted pipeline to generate predictions for this new data.
- Build a new data frame called containing the following columns:
  - `Info_PepID` (taken from df_holdout)
  - `Info_pos` (taken from df_holdout)
  - `Predicted_class` (with the predictions produced by your pipeline)
- Save your predictions dataframe to a file called **mypreds.rds**. You will be able to submit this file, and it will be used to give you some feedback on the actual performance of your pipeline on new data.

```r
holdout_preds <- predict(final_rf_fit, new_data = df_holdout)
mypreds <- df_holdout %>%
  select(Info_PepID, Info_pos) %>%
  mutate(Predicted_class = holdout_preds$.pred_class)

saveRDS(mypreds, "mypreds.rds")
```

The final model was successfully applied to the unlabeled holdout data and the predictions were saved to file mypreds.rds.

=========== End of Task II ===========