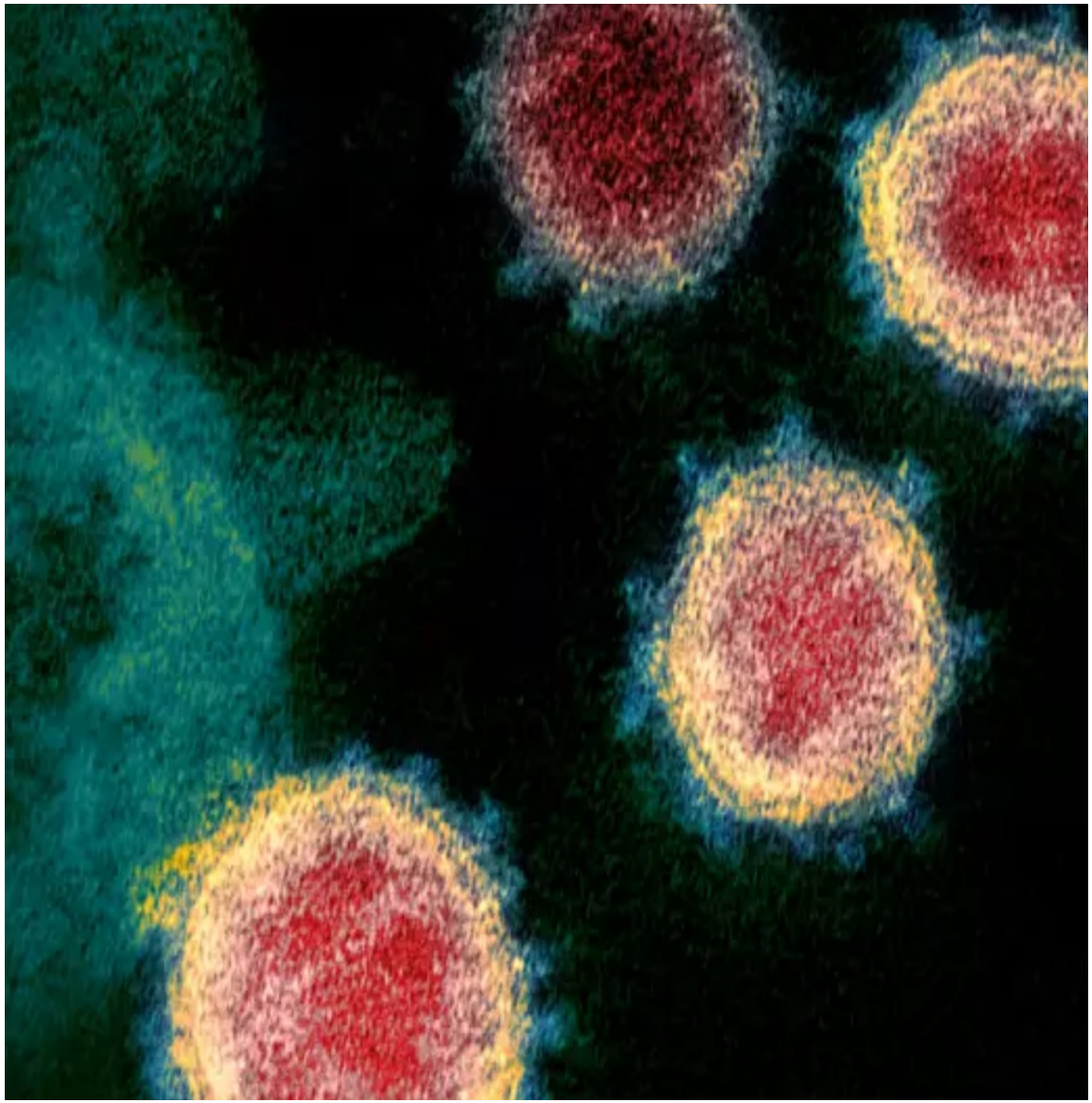
<Murat Kucukosmanoglu>

<10/08/2022>



Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusions and Recommendations

Executive Summary

A. Summary of methodologies

1. Data Collection from the laboratory tests.
2. Data pre-processing
3. Exploratory Data Analysis
4. Imputation and Feature Engineering
5. Predictive Models (Classification)

B. Summary of all results

1. We built and tested four machine learning models: logistic regression, decision tree, Support-vector machine and Random Forest Model to predict whether the patient is positive/negative for Covid 19
2. Based on the accuracy and recall of the train data, the Support-vector machine is predicted to have been the best model.

Introduction

A. Project background and context

1. The exponential rise in the number of COVID cases threatened to overwhelm health systems worldwide with a demand for hospitalization and for ICU beds far above the capacity.

2. Testing every case with some mild symptoms, would be impractical in the context of an overburdened health system with the potential limitation to performing tests for the detection of SARS-CoV-2

B. Problems you want to find answers

1. Based on the laboratory tests collected from the suspected cases, improve a classification model that predicts the chances of being positive/negative for covid19

2. What features could influence in the acceptance of patient into hospital?

DATA REPORT

Executive Summary

Data collection methodology:

Data Collected from Hospital Israelita Albert Einstein APIs.

Visual inspection of data

Rows, Columns and descriptive analysis of data

Understanding of Attributes (variable info and rename)

Combine and clean the edited data for future usage and analysis..

Data Report

- **Data collection methodology:**
  - Data Collected from Hospital Israelita Albert Einstein APIs.
- **Visual inspection of data**
  - Rows, Columns and descriptive analysis of data
  - Understanding of Attributes (variable info and rename)
  - Combine and clean the edited data for future usage and analysis.

Data Collection

- **Describe how data sets were collected.**

The data was gathered using a combination of API queries from Hospital Israelita Albert Einstein. There are 5644 rows and 111 columns. The summary of the dataframe for nine column is shown below.

Hospital Israelita Albert Einstein API

Patient ID	Patient age quantile	SARS-Cov-2 exam result	Patient addmited to regular ward (1=yes, 0=no)	Patient addmited to semi-intensive unit (1=yes, 0=no)	Patient addmited to intensive care unit (1=yes, 0=no)	Hematocrit	Hemoglobin	Platelets

Data Collection

- **Patient ID and empty columns are removed as these columns will not be a factor at all.**
- **It is best to write column names in lower case and leave no space between them so that users may select features with easily.**

```
columns = []
for column in df.columns:
columns.append(column.lower().replace(' ','_'))

# Assign the new columns into current dataframe.
df.columns = columns
```

Data Collection

- # Now check the renamed column names
- df.head()

	patient_age_quantile	sars-cov-2_exam_result	patient_addmited_to_regular_ward_(1=yes,_0=no)	patient_addmited_to_semi-intensive_unit_(1=yes,_0=no)
0	13	negative	0	0
1	17	negative	0	0
2	8	negative	0	0
3	5	negative	0	0
4	15	negative	0	0

Data pre-processing (Discrete Features)

- Distribution and spread for both categorical and continuous variable has to be defined.
- Therefore, dataframe has to be defined as categorical attribute and continuous attribute. We start with categorical variable.

Data pre-processing (Discrete Features)

- There are 42 categorical values with variable 5 and less than 5.

```
# We first need to look at the categorical attributes.
categ = []
# Loop to evaluate if it's categorical
for j in df.columns:
    if len(df[j].unique()) <= 5:
        categ.append(j)
print(categ)
len(categ)
```

0.1s Python

```
['sars-cov-2_exam_result', 'patient_addmited_to_regular_ward_(1=yes,_0=no)', 'patient_addmited_to_semi-intensive_unit_(1=yes,_0=no)',
'patient_addmited_to_intensive_care_unit_(1=yes,_0=no)', 'respiratory_syncytial_virus', 'influenza_a', 'influenza_b', 'parainfluenza_1',
'coronavirusnl63', 'rhinovirus/enterovirus', 'coronavirus_hku1', 'parainfluenza_3', 'chlamydophila_pneumoniae', 'adenovirus', 'parainfluenza_4',
'coronavirus229e', 'coronavirusoc43', 'inf_a_h1n1_2009', 'bordetella_pertussis', 'metapneumovirus', 'parainfluenza_2', 'influenza_b,_rapid_test',
'influenza_a,_rapid_test', 'strepto_a', 'fio2_(venous_blood_gas_analysis)', 'promyelocytes', 'metamyelocytes', 'myelocytes', 'myeloblasts',
'urine_-_esterase', 'urine_-_aspect', 'urine_-_hemoglobin', 'urine_-_bile_pigments', 'urine_-_ketone_bodies', 'urine_-_nitrite', 'urine_-
_urobilinogen', 'urine_-_protein', 'urine_-_hyaline_cylinders', 'urine_-_granular_cylinders', 'urine_-_yeasts', 'urine_-_color', 'vitamin_b12']

42
```

Category analysis for all discrete features

- Seven features have only one category besides the NaN category.
- Because they will not affect prediction, they are removed from the data frame.
- Now, 35 discrete features remain.

```
# Eval categories for all discrete features
categ1 = categ[1:20]
for i in categ1:
    print(i, ' - ', df[i].unique())
```

✓ 0.9s

```
patient_addmited_to_regular_ward_(1=yes,_0=no)  -  [0 1]
patient_addmited_to_semi-intensive_unit_(1=yes,_0=no)  -  [0 1]
patient_addmited_to_intensive_care_unit_(1=yes,_0=no)  -  [0 1]
respiratory_syncytial_virus  -  [nan 'not_detected' 'detected']
influenza_a  -  [nan 'not_detected' 'detected']
influenza_b  -  [nan 'not_detected' 'detected']
parainfluenza_1  -  [nan 'not_detected' 'detected']
coronavirusnl63  -  [nan 'not_detected' 'detected']
rhinovirus/enterovirus  -  [nan 'detected' 'not_detected']
coronavirus_hku1  -  [nan 'not_detected' 'detected']
parainfluenza_3  -  [nan 'not_detected' 'detected']
chlamydophila_pneumoniae  -  [nan 'not_detected' 'detected']
adenovirus  -  [nan 'not_detected' 'detected']
parainfluenza_4  -  [nan 'not_detected' 'detected']
coronavirus229e  -  [nan 'not_detected' 'detected']
coronavirusoc43  -  [nan 'not_detected' 'detected']
inf_a_h1n1_2009  -  [nan 'not_detected' 'detected']
bordetella_pertussis  -  [nan 'not_detected' 'detected']
metapneumovirus  -  [nan 'not_detected' 'detected']
```

Missing values treatment

- There are several missing values, ranging from %76 to %100.
- Missing values of more than **90%** were removed. **As a result, 22 columns were left.**

```
percent_missing = df1.isnull().sum() * 100 / len(df1)
missing_value_df1 = pd.DataFrame({'column_name': df1.columns,
                                  'percent_missing': percent_missing})
missing_value_df1.sort_values('percent_missing', inplace=True)
missing_value_df1.head(50)
```

✓ 0.1s

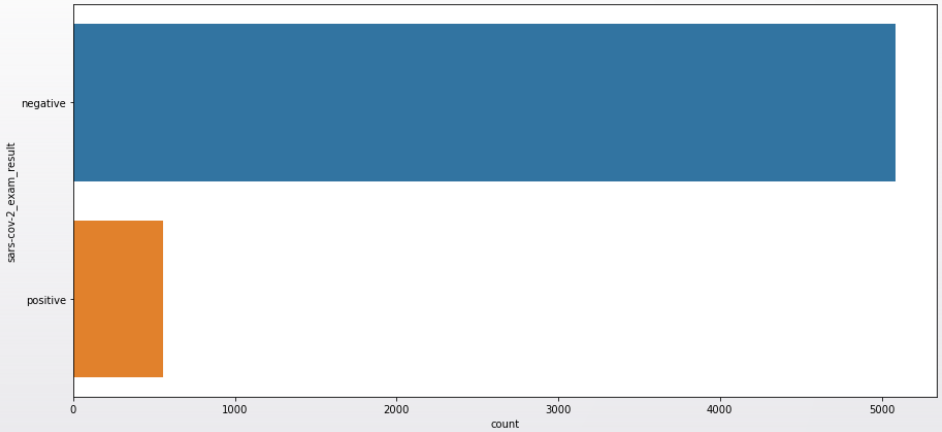
	column_name	percent_missing
sars-cov-2_exam_result	sars-cov-2_exam_result	0.000000
patient_addmited_to_regular_ward_(1=yes,_0=no)	patient_addmited_to_regular_ward_(1=yes,_0=no)	0.000000
patient_addmited_to_semi-intensive_unit_(1=yes,_0=no)	patient_addmited_to_semi-intensive_unit_(1=yes...	0.000000
patient_addmited_to_intensive_care_unit_(1=yes,_0=no)	patient_addmited_to_intensive_care_unit_(1=yes...	0.000000
respiratory_syncytial_virus	respiratory_syncytial_virus	76.009922
influenza_a	influenza_a	76.009922
influenza_b	influenza_b	76.009922
metapneumovirus	metapneumovirus	76.045358
bordetella_pertussis	bordetella_pertussis	76.045358
coronavirusoc43	coronavirusoc43	76.045358
coronavirus229e	coronavirus229e	76.045358
parainfluenza_4	parainfluenza_4	76.045358

Exploratory Data Analysis (Discrete Features)

- Uni-variate analysis (distribution of data in categories for categorical ones)
- Bi-variate analysis (relationship between different variables, correlations).

Distribution of the Variables

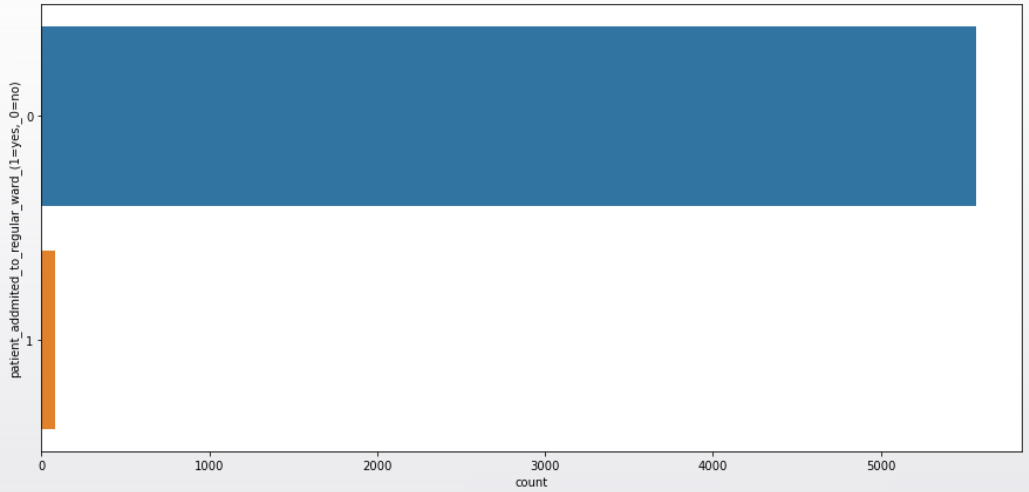
- First, the number of cases of COVID reported as both positive and negative were analyzed.




```
negative    5086
positive     558
Name: sars-cov-2_exam_result, dtype: int64
```

Distribution of the Variables

- Second, the number of patients admitted to the regular ward was analyzed.



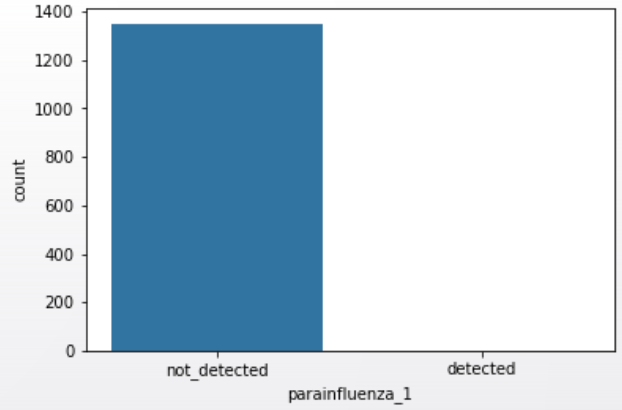

```
df["patient_addmited_to_regular_ward_(1=yes,_0=no)"].value_counts()
```

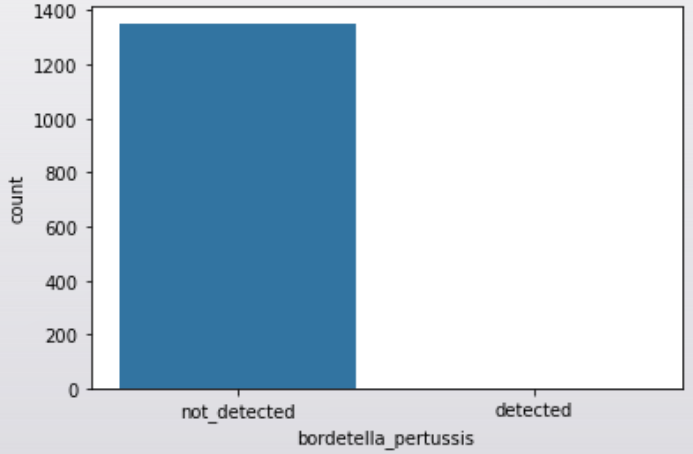
✓ 0.8s

```
0    5565
1      79
Name: patient_addmited_to_regular_ward_(1=yes,_0=no), dtype: int64
```

Distribution of the Variables

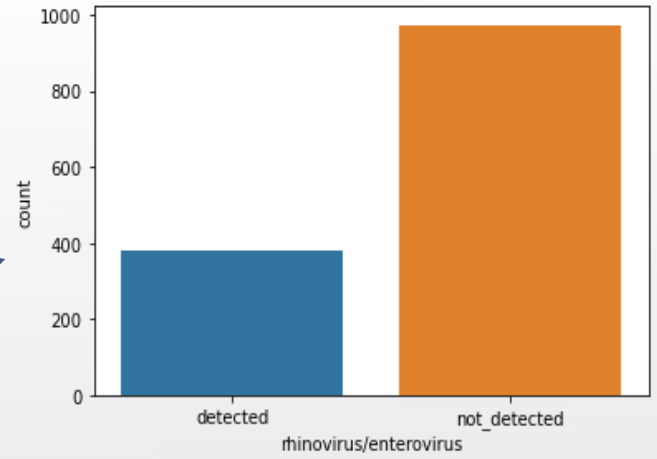
- Lastly, all the discrete features were analyzed.
- There is almost no positive case in bordetella and parafiluenza_1



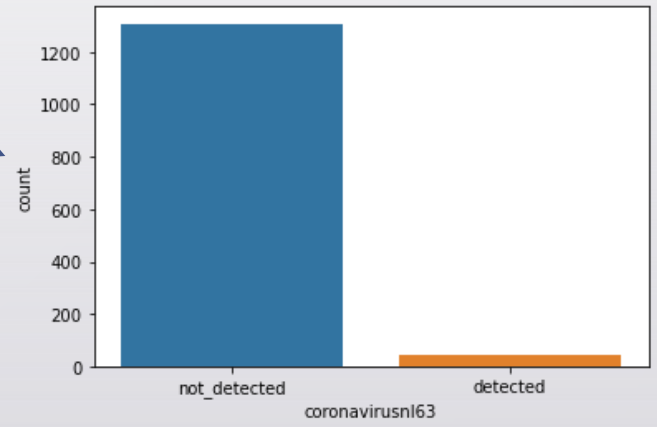


Distribution of the Variables

- The distribution of rhinoviruses and enteroviruses between the two groups is "relatively" more equal.



- Negative cases dominate significantly for other categories.



Bivariate Distributions

- Exploratory Data Analysis performed on both categorical and continuous variables. Here focus will be categorical features.
- A stripplot is a fantastic way to demonstrate which attributes are associated with positive sars-cov-2_exam_result.
- COVID results were invariably negative when some attributes such as the influenza, metapneumovirus, parainfluenza and adenovirus test were positive. These characteristics may serve as excellent indicators of a patient's absence of a COVID positive case.

Data pre-processing (Continuous Features)

```
# Basic summary stats – Continuous  Features
df2.describe().T
```

✓ 0.2s Python

	count	mean	std	min	25%	50%	75%	max
ferritin	23.0	7.288611e-09	1.022475	-0.627529	-0.559897	-0.358466	0.119507	3.845748
arteiral_fio2	20.0	4.656613e-09	1.025978	-1.532932	-0.121498	-0.011744	-0.011744	2.841856
basophils	602.0	-6.633740e-09	1.000832	-1.140144	-0.529226	-0.223767	0.387152	11.078219
mean_platelet_volume_	599.0	7.438142e-09	1.000836	-2.457575	-0.662483	-0.101517	0.683835	3.713052
vitamin_b12	3.0	-1.986821e-08	1.224745	-1.400606	-0.434897	0.530811	0.700303	0.869795
...	...	...	...	...	...	...	...	...
lipase_dosage	8.0	-3.725290e-09	1.069045	-1.192227	-0.547022	-0.350655	0.182341	1.725222
base_excess_(venous_blood_gas_analysis)	136.0	-1.075130e-09	1.003697	-3.669449	-0.402065	0.080306	0.553576	3.356791
pco2_(arterial_blood_gas_analysis)	27.0	8.416397e-09	1.019049	-1.244817	-0.534810	-0.212080	0.023052	3.236524
mean_corpuscular_hemoglobin_concentration (mchc)	602.0	1.014863e-09	1.000832	-5.431808	-0.552476	-0.054585	0.642463	3.331071
arterial_lactic_acid	27.0	-1.655685e-09	1.019049	-1.091068	-0.694761	-0.298454	0.229956	3.004107

66 rows × 8 columns

Missing values treatment

- There are several missing values, ranging from %89 to %100.

- Missing values of more than **90%** were removed. **As a result, 15 columns were left.**

	column_name	percent_missing
patient_age_quantile	patient_age_quantile	0.000000
hemoglobin	hemoglobin	89.316088
hematocrit	hematocrit	89.316088
basophils	basophils	89.333806
lymphocytes	lymphocytes	89.333806
mean_corpuscular_hemoglobin_concentration (mchc)	mean_corpuscular_hemoglobin_concentration (mchc)	89.333806
platelets	platelets	89.333806
red_blood_cells	red_blood_cells	89.333806
leukocytes	leukocytes	89.333806
red_blood_cell_distribution_width_(rdw)	red_blood_cell_distribution_width_(rdw)	89.333806
eosinophils	eosinophils	89.333806
mean_corpuscular_volume_(mcv)	mean_corpuscular_volume_(mcv)	89.333806
mean_corpuscular_hemoglobin_(mch)	mean_corpuscular_hemoglobin_(mch)	89.333806
monocytes	monocytes	89.351524
mean_platelet_volume_	mean_platelet_volume_	89.386960
neutrophils	neutrophils	90.910702
proteina_c_reativa_mg/dl	proteina_c_reativa_mg/dl	91.034727
creatinine	creatinine	92.487597
urea	urea	92.965982
potassium	potassium	93.426648
sodium	sodium	93.444366

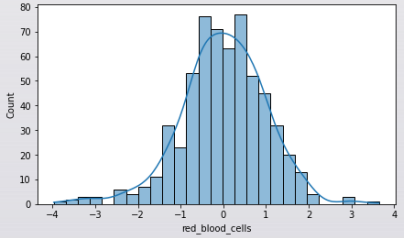
Exploratory Data Analysis (Continuous features)

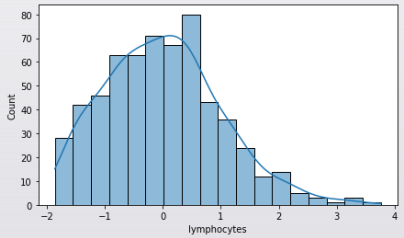
- Uni-variate analysis (distribution and spread for every continuous attribute)
- Bi-variate analysis (relationship between different variables, correlations)

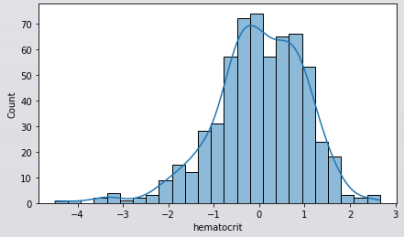
Distribution for continuous features

- More of the continuous features are close to normal distribution. For example, mean_platet, red blood_cells, monocytes etc.

- However, some of them are right, and some of them are left skewed. As an example, lymphocytes is right skewed, and hematocrit is left-skewed

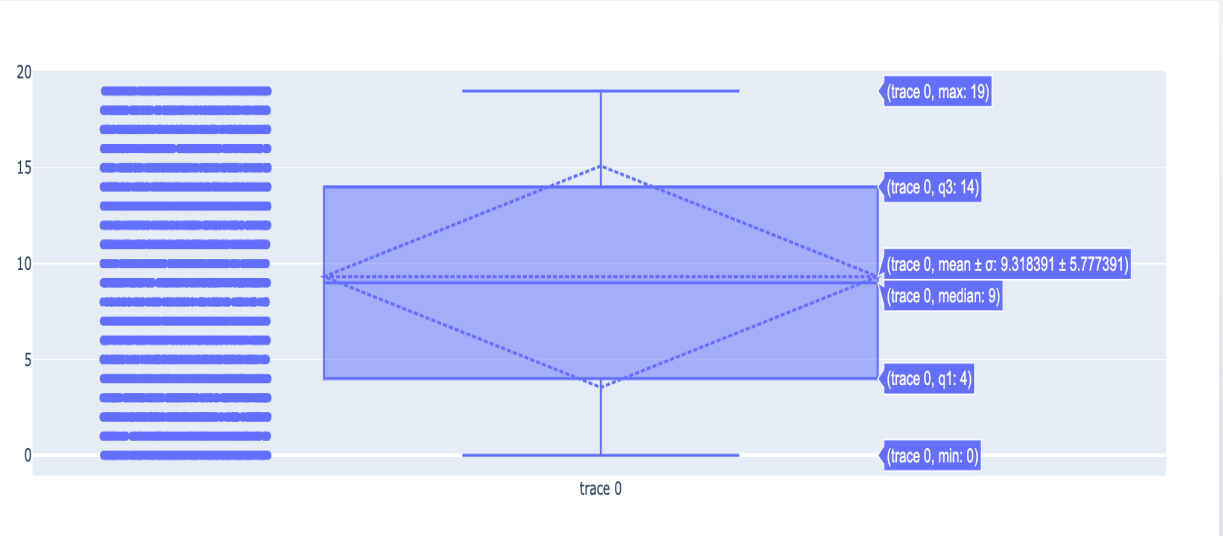






Patient age

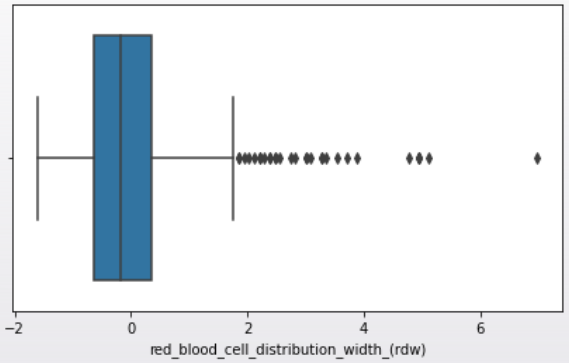
- In the patient age quantile, the Median is 9 and the mean is 9.31 +/ 5.77. There seem to be no outliers at this point.

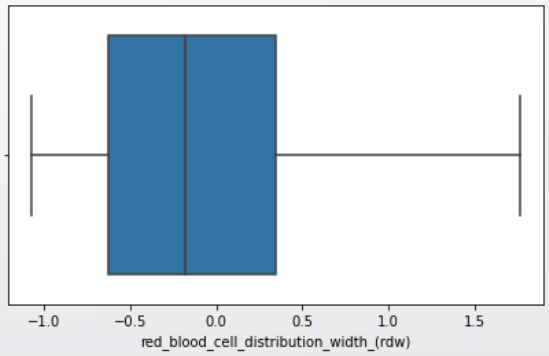


Outlier Treatment

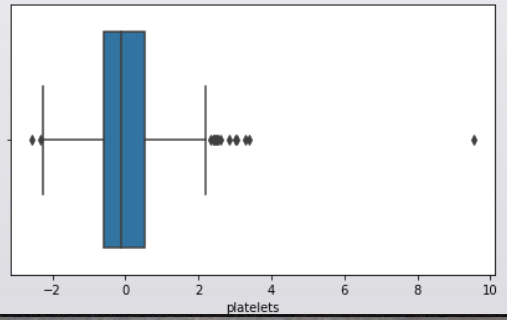
- There are a few outliers in each numerical features, except for patient age. I decided the remove the data with 95% confidence level.

```
# remove the data out of 95% confidence level
def filter_data(df):
    for col in df.columns:
        print("capping the ",col)
        if (((df[col].dtype)=='float64') | ((df[col].dtype)=='int64')):
            percentiles = df[col].quantile([0.025,0.975]).values
            df[col][df[col] <= percentiles[0]] = percentiles[0]
            df[col][df[col] >= percentiles[1]] = percentiles[1]
        else:
            df[col]=df[col]
    return df
```

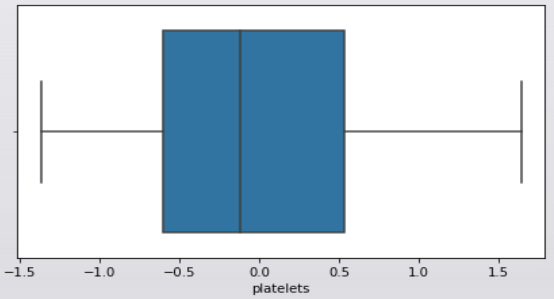




Before



After

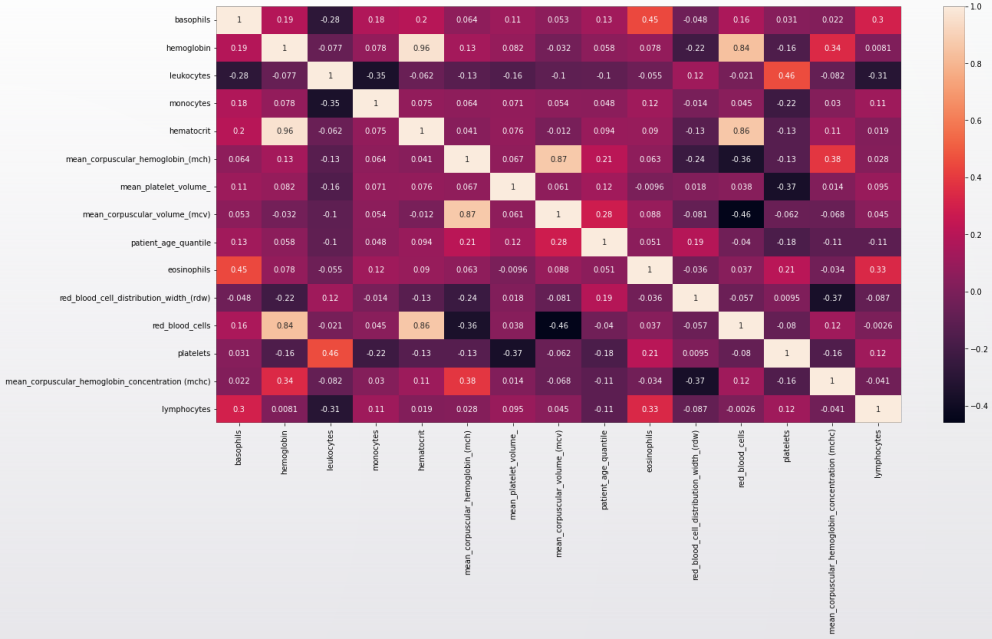


Correlation between features

- Red blood cells and haemoglobin have a positive correlation of 0.86. Haemoglobin, a protein rich in iron that gives blood its red color, is found in red blood cells.

- Hematocrit and haemoglobin have a positive correlation of 0.96. Red blood cell measures like haemoglobin and hematocrit identify dietary deficits, acute diseases, and long-term medical disorders.

`Not`: We shall be careful about the collinearity in statistics. When predictor variables in the same regression model are highly-correlated, they cannot independently predict the value of the dependent variable.



Abs values higher than 0.85 must be taken into account for collinearity. I want to do necessary analysis after imputing. By doing this, we will see final form of data to be trained!

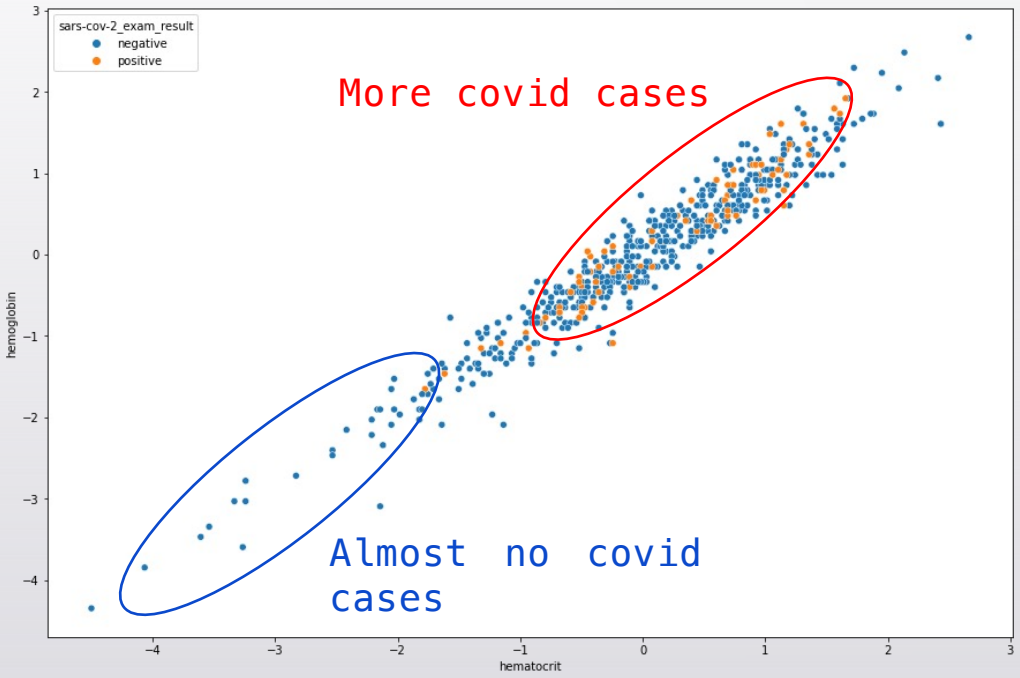
hemoglobin and hematocrit and red_blood_cells mean_corpuscular_volume_(mcv) and mean_corpuscular_hemoglobin_(mch)

Bivariate Distributions

- To show how these points connect to the positive and negative COVID cases, I wanted to highlight them in this Figure.

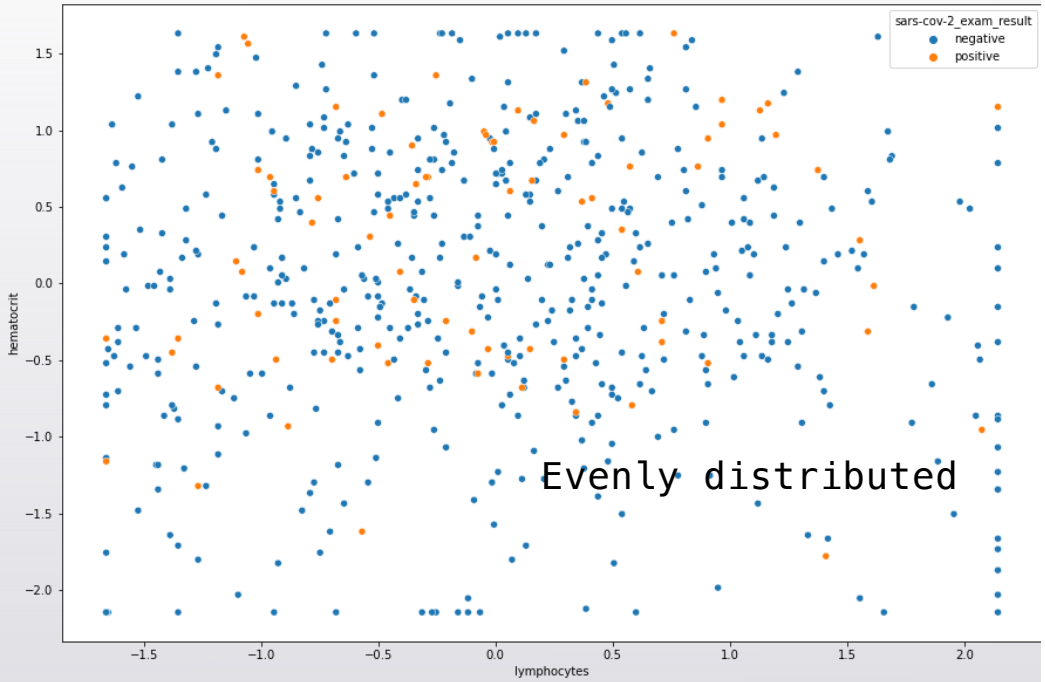
Hematocrit and Hemoglobin

- Hematocrit and hemoglobin are two features where positive covid instances exist in their positive values. These features may signal positive COVID scenarios when we have greater values for them.
- No indication of COVID at certain areas.



Lymphocytes and Hematocrit

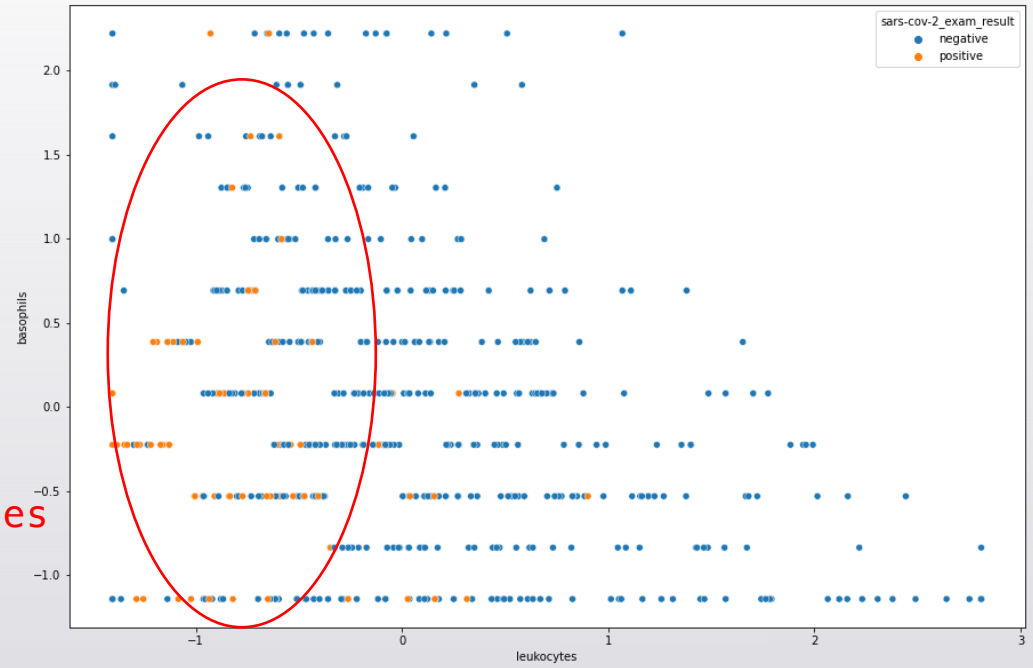
- These are two examples of points where both positive and negative covid instances are evenly distributed along the points.
- No indication of COVID at certain areas.



Basophils and Leukocytes

- Basophils and leukocytes are not well correlated, but positive covid cases exist more in the negative leukocyte's values.

More covid cases



Imputation and Feature Engineering

- Changing String values to numerical values
- KNN Imputation
- Removing Multi-collineratiy

Changing String values to numerical values

- In discrete values, variables such as positive, detected were made into a value of 1.
- Variables such as negative and not-detected were made into 0.
- Continuous variables are not required to carry out this transformation.

```
# Changing string values to numerical values
for j in categ:
    if 'positive' in list(df1[j].unique()):
        df1[j].replace('positive',1, inplace=True)
    elif 'detected' in list(df1[j].unique()):
        df1[j].replace('detected',1, inplace=True)
    if 'negative' in list(df1[j].unique()):
        df1[j].replace('negative',0, inplace=True)
    elif 'not_detected' in list(df1[j].unique()):
        df1[j].replace('not_detected',0, inplace=True)
```

✓ 0.8s

Combining the discrete and continuous variable

```
# Combine the two datasets
df_all = pd.concat([df1, df2], axis=1)
df_all.head()
```

We have 37 columns in total, and 5 of them has no missing values.

- **In order to increase the accuracy of the model, I decided to perform model on the rows with fewer than 50% missing values. Following a literature study, I arrived at this conclusion. Thus, 366 rows and 37 columns remained.**

KNN Imputation

- **KNN imputation** is one of the imputation methods that can be used for this. This sklearn method requires us to encode categorical variables if we are using them for imputation. In this case, we'll use only selected numeric features for imputation.
- Imputed variables are added to original dataset. No missing values are left.

```
from sklearn.impute import KNNImputer
imputer = KNNImputer(n_neighbors = 2, weights = "uniform")  # 2 Nearest Neighbours
temp_df_for_imputation = categ_missing

temp_df_for_imputation = imputer.fit_transform(df_all[temp_df_for_imputation])
temp_df_for_imputation = pd.DataFrame(
    temp_df_for_imputation,
    columns=categ_missing,
)
```

✓ 0.2s

+ Code + Markdown

```
temp_df_for_imputation.columns
```

✓ 0.6s

```
Index(['respiratory_syncytial_virus', 'influenza_a', 'influenza_b',
       'parainfluenza_1', 'coronavirusnl63', 'rhinovirus/enterovirus',
       'coronavirus_hku1', 'parainfluenza_3', 'chlamydophila_pneumoniae',
       'adenovirus', 'parainfluenza_4', 'coronavirus229e', 'coronavirusoc43',
       'inf_a_h1n1_2009', 'bordetella_pertussis', 'metapneumovirus',
       'influenza_b,_rapid_test', 'influenza_a,_rapid_test', 'leukocytes',
       'hemoglobin', 'red_blood_cell_distribution_width_(rdw)', 'eosinophils',
       'mean_corpuscular_hemoglobin_(mch)', 'platelets',
       'mean_corpuscular_hemoglobin_concentration (mchc)', 'monocytes',
       'mean_platelet_volume_', 'hematocrit', 'red_blood_cells',
       'mean_corpuscular_volume_(mcv)', 'lymphocytes', 'patient_age_quantile'],
      dtype='object')
```

```
Output exceeds the size limit. Open the full output data in a text editor
sars-cov-2_exam_result                                  0
patient_addmited_to_regular_ward_(1=yes,_0=no)          0
patient_addmited_to_semi-intensive_unit_(1=yes,_0=no)   0
patient_addmited_to_intensive_care_unit_(1=yes,_0=no)   0
respiratory_syncytial_virus                             0
influenza_a                                             0
influenza_b                                             0
parainfluenza_1                                         0
coronavirusnl63                                         0
rhinovirus/enterovirus                                  0
coronavirus_hku1                                        0
parainfluenza_3                                         0
chlamydophila_pneumoniae                                0
adenovirus                                              0
parainfluenza_4                                         0
coronavirus229e                                         0
coronavirusoc43                                         0
inf_a_h1n1_2009                                         0
bordetella_pertussis                                    0
metapneumovirus                                         0
influenza_b,_rapid_test                                 0
influenza_a,_rapid_test                                 0
platelets                                               0
lymphocytes                                             0
mean_corpuscular_volume_(mcv)                           0
...
red_blood_cells                                         0
monocytes                                               0
```

Removing Multicollinearity

- **Two datasets were combined.**

```
# Combine the two datasets
df_all = pd.concat([df1, df2], axis=1)
df_all.head()
```

- ****To remove multicollinearity****
- 1. Drop every column that has VIF score greater than 5, one by one
- 2. Look at the adjusted R square of all these models
- 3. Drop the Variable that makes the least change in Adjusted-R square
- 4. Check the VIF Scores again
- 5. Continue till you get all VIF scores under 5

```
import statsmodels.api as sm
om statsmodels.stats.outliers_influence import variance_inflation_factor

X = df_all[list(df_all.columns[1:])]
vif_info = pd.DataFrame()
vif_info['VIF'] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
vif_info['Column'] = X.columns
vif_info.sort_values('VIF', ascending=False)
```

✓ 7.6s

	VIF	Column
31	245.677334	hematocrit
22	207.932085	hemoglobin
26	117.280896	mean_corpuscular_hemoglobin_(mch)
33	112.337391	mean_corpuscular_volume_(mcv)
32	42.175923	red_blood_cells
28	31.171306	mean_corpuscular_hemoglobin_concentration (mchc)

6 features were found to be higher than 5. They are shown in the table.

Feature Engineering

- From those features, we can generate the new one. The new feature will contain the addition of those pairs. After we create those features, we can safely remove them from our data.

```
X["hem_add"]= X['hemoglobin']+X['hematocrit']
X['corpuslar_add']= X['mean_corpuscular_volume_(mcv)']+X['mean_corpuscular_hemoglobin_(mch)']
X = X.drop(['hemoglobin','hematocrit','mean_corpuscular_volume_(mcv)','mean_corpuscular_hemoglobin_(mch)'], axis=1)
#X=df_all.drop(['hemoglobin','mean_corpuscular_hemoglobin_(mch)','full_count'], axis=1)
X.head()
```

- As you can see from the example above, we still have variables with very high VIF values. Nevertheless, creating new features had a wonderful outcome for us.

```
vif_info = pd.DataFrame()
vif_info['VIF'] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
vif_info['Column'] = X.columns
vif_info.sort_values('VIF', ascending=False)
```

✓ 0.2s

	VIF	Column
29	35.881002	red_blood_cells
32	31.142053	hem_add
33	9.192266	corpuslar_add

Feature Engineering

- **Nice! Currently, every variable has a VIF value of less than 5. Now that we have those variables, we can interpret the outcome. Let's first create our machine learning model!**

```
X = X.drop(["hem_add"], axis=1)
```

```
vif_info = pd.DataFrame()
vif_info['VIF'] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
vif_info['Column'] = X.columns
vif_info.sort_values('VIF', ascending=False)
```

	VIF	Column
30	2.379347	leukocytes
23	2.183216	patient_age_quantile
21	1.973136	platelets
22	1.849858	lymphocytes
32	1.813986	corpuslar_add
28	1.570995	red_blood_cells

Predictive Analysis (Classification)

Proposed approach

Potential techniques :

Since it is a binary prediction problem, we will use classification models logistic regression, SVM, Decision Tree and Random Forest. One of the most used analytical methods, binary prediction has several uses across a variety of industries. Because logistic regression is highly descriptive with good accuracy and has reasonable computational requirements , I will start with it.

Building the model

We will be building 4 different models:

- **Logistic Regression**
- **Support Vector Machine(SVM)**
- **Decision Tree**
- **Random Forest**

Overall design :

- create a column for the class
- Standardize the data
- Splitting the data and proceeding with modeling.
- Find best Hyperparameter for SVM, Classification Trees and Logistic Regression
- Find the method performs best using test data

Predictive Analysis (Classification)

- **We have total of 366 cases. 52 of them is positive, and 314 of them is negative!**
- **Splited the data into 80% train and 20% test set**

```
## Splitting the dataset into the Training set and Test set
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=1,stratify=y)
```

- **We have 292 y_train data with 251 negative and 41 positive result!**
- **We have 63 y_test data with 63 negative and 11 positive result!**

```
y_train.value_counts()
0.1s
0    251
1     41
Name: sars-cov-2_exam_result, dtype: int64

y_test.value_counts()
0.2s
0    63
1    11
Name: sars-cov-2_exam_result, dtype: int64
```

RESULTS

1. Logistic Regression Model
2. Support Vector Machine
3. Decision Tree
4. Random Forest

Logistic Regression Model

- **We are getting an accuracy of around 93%** on train and test dataset.
- On the train and test datasets, however, **the recall** for this model is only about 64% for class 1.
- Due to the low recall, **this model will not perform well** in identifying individuals who have a high likelihood of testing positively for COVID. As a result, it will be difficult to predict if a patient will get COVID.

```
# Checking the performance on the training data

y_pred_train = lg.predict(X_train)
metrics_score(y_train, y_pred_train)
```

✓ 0.2s

	precision	recall	f1-score	support
0	0.94	1.00	0.97	251
1	0.96	0.63	0.76	41
accuracy			0.95	292
macro avg	0.95	0.82	0.87	292
weighted avg	0.95	0.95	0.94	292

```
# Checking the performance on the test dataset
y_pred_test = lg.predict(X_test)
metrics_score(y_test, y_pred_test)
```

✓ 0.2s

	precision	recall	f1-score	support
0	0.94	0.98	0.96	63
1	0.88	0.64	0.74	11
accuracy			0.93	74
macro avg	0.91	0.81	0.85	74
weighted avg	0.93	0.93	0.93	74

Logistic Regression Model

- **We are getting an accuracy of around 93%** on train and test dataset.
- On the train and test datasets, however, **the recall** for this model is only about 64% for class 1.
- Due to the low recall, **this model will not perform well** in identifying individuals who have a high likelihood of testing positively for COVID. As a result, it will be difficult to predict if a patient will get COVID.

```
# Checking the performance on the training data

y_pred_train = lg.predict(X_train)
metrics_score(y_train, y_pred_train)
```

✓ 0.2s

	precision	recall	f1-score	support
0	0.94	1.00	0.97	251
1	0.96	0.63	0.76	41
accuracy			0.95	292
macro avg	0.95	0.82	0.87	292
weighted avg	0.95	0.95	0.94	292

```
# Checking the performance on the test dataset
y_pred_test = lg.predict(X_test)
metrics_score(y_test, y_pred_test)
```

✓ 0.2s

	precision	recall	f1-score	support
0	0.94	0.98	0.96	63
1	0.88	0.64	0.74	11
accuracy			0.93	74
macro avg	0.91	0.81	0.85	74
weighted avg	0.93	0.93	0.93	74

Logistic Regression Model

- **Let's look at the observations that we got from the coefficients of logistic regression model**

Observations:

Features which positively affect on the COVID's positive case are:

- patient_addmited_to_regular_ward_(1=yes,_0=no)
- patient_addmited_to_intensive_care_unit_(1=yes,_0=no)
- red_blood_cells
- patient_addmited_to_semi-intensive_unit_(1=yes,_0=no)
- monocytes
- corpuslar_add
- influenza_b,_rapid_test
- patient_age_quantile

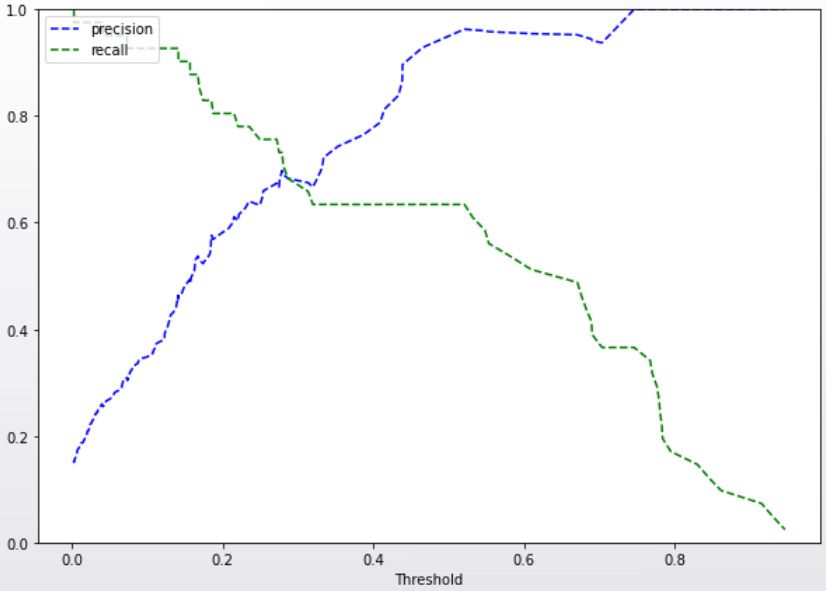
Features which negatively affect on the COVID's positive case are:

- rhinovirus/enterovirus
- inf_a_h1n1_2009
- leukocytes
- influenza_b
- eosinophils
- platelets
- influenza_a,_rapid_test
- parainfluenza_3
- coronavirusoc43
- influenza_a
- coronavirus_hku1

Precision-Recall Curve for Logistic Regression

- We can see that precision and recall are balanced for a threshold of about **0.3133**.

- Let's find out the performance of the model at this threshold*




```
optimal_threshold=thresholds_lg[i]
y_pred_train = lg.predict_proba(X_train)
metrics_score(y_train, y_pred_train[:,1]>optimal_threshold)
```

✓ 1.4s

	precision	recall	f1-score	support
0	0.94	0.95	0.94	251
1	0.67	0.63	0.65	41
accuracy			0.90	292
macro avg	0.80	0.79	0.80	292
weighted avg	0.90	0.90	0.90	292

```
y_pred_test = lg.predict_proba(X_test)
metrics_score(y_test, y_pred_test[:,1]>optimal_threshold)
```

✓ 0.3s

	precision	recall	f1-score	support
0	0.97	0.98	0.98	63
1	0.90	0.82	0.86	11
accuracy			0.96	74
macro avg	0.93	0.90	0.92	74
weighted avg	0.96	0.96	0.96	74

- The model performance has improved. The recall has increased significantly for class 1 from 0.66 to 0.82 on test set. However, its being much higher than training's model performance is questionable!

Support Vector Machine (Linear Kernel)

Recall is low with Linear Kernel, therefore, we will try a non-linear kernel.

Linear Kernel

```
# Fitting SVM
svm = SVC(kernel='linear') # Linear kernal or linear decision boundary
model = svm.fit(X= X_train_scaled, y = y_train)
```

✓ 0.1s

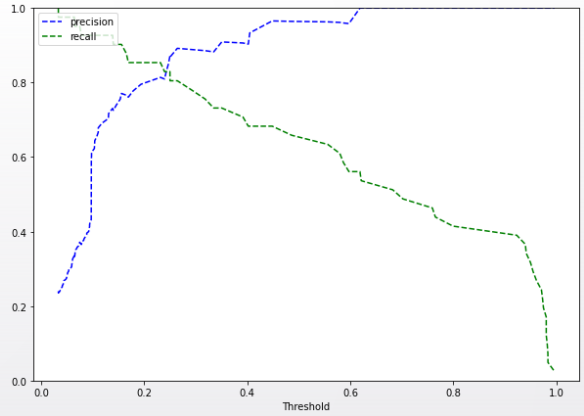
```
y_pred_train_svm = model.predict(X_train_scaled)
metrics_score(y_train, y_pred_train_svm)
```

✓ 1.1s

	precision	recall	f1-score	support
0	0.93	0.98	0.96	251
1	0.85	0.56	0.68	41
accuracy			0.92	292
macro avg	0.89	0.77	0.82	292
weighted avg	0.92	0.92	0.92	292

Support Vector Machine (RBF Kernel)

- **Recall significantly improved on train dataset with RBF Kernel with threshold of 0.247. Let's see how it performs on test dataset.**



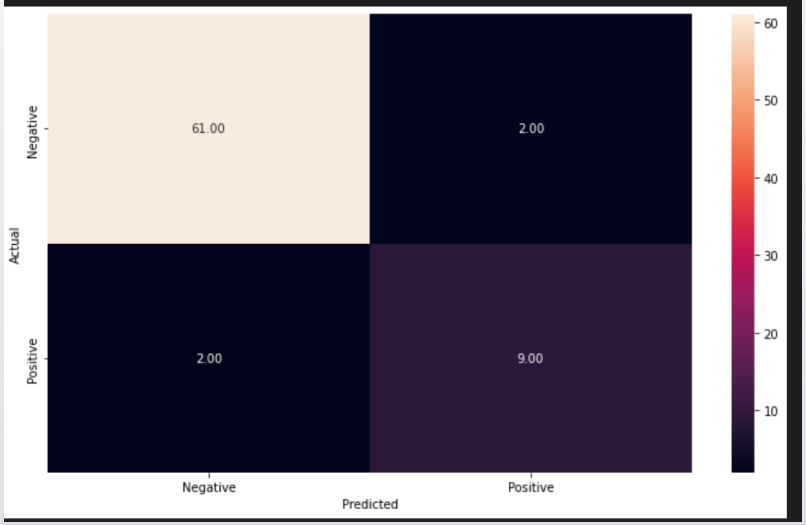

```
optimal_threshold1=thresholds_svm[i]
y_pred_train = svm_rbf.predict_proba(X_train_scaled)

metrics_score(y_train, y_pred_train[:,1]>optimal_threshold1)
```

✓ 1.4s

	precision	recall	f1-score	support
0	0.97	0.98	0.98	251
1	0.87	0.83	0.85	41
accuracy			0.96	292
macro avg	0.92	0.90	0.91	292
weighted avg	0.96	0.96	0.96	292

Support Vector Machine (RBF Kernel)

	precision	recall	f1-score	support
0	0.97	0.97	0.97	63
1	0.82	0.82	0.82	11
accuracy			0.95	74
macro avg	0.89	0.89	0.89	74
weighted avg	0.95	0.95	0.95	74



****Observations:****

* At the optimal threshold of .247, the model performance has performed greatly on test dataset.
* Moreover, the kernel used to create this is rbf, hence model is performing good with non-linear kernel.
* As the recall is good, ****this model will perform well**** in predicting if the patient's covid case.

Decision Tree

- **For every statistic on the training dataset, the decision tree is returning a 100% score.**

- **Clearly, the train dataset has been overfit.**

- **Because of this, it does not perform well on the test dataset. With 36% percentage, hospitals may be unable to predict positive cases, which is extremely risky for the general public's health and hospital's business.**

```
# Checking performance on the training dataset
y_train_pred_dt = dt.predict(X_train)

metrics_score(y_train, y_train_pred_dt)
```

✓ 0.2s

	precision	recall	f1-score	support
0	1.00	1.00	1.00	251
1	1.00	1.00	1.00	41
accuracy			1.00	292
macro avg	1.00	1.00	1.00	292
weighted avg	1.00	1.00	1.00	292

```
# Checking performance on the test dataset
y_test_pred_dt = dt.predict(X_test)

metrics_score(y_test, y_test_pred_dt)
```

✓ 0.2s

	precision	recall	f1-score	support
0	0.94	0.94	0.94	63
1	0.64	0.64	0.64	11
accuracy			0.89	74
macro avg	0.79	0.79	0.79	74
weighted avg	0.89	0.89	0.89	74

Decision Tree (Tuned)

- **To overcome the issue of overfitting, I decided to use the tuned model with default values.**

```
# Checking performance on the training dataset
y_train_pred_dt = dtree_estimator.predict(X_train)
metrics_score(y_train, y_train_pred_dt)
```

✓ 1.1s

	precision	recall	f1-score	support
0	0.97	0.78	0.87	251
1	0.39	0.85	0.54	41
accuracy			0.79	292
macro avg	0.68	0.82	0.70	292
weighted avg	0.89	0.79	0.82	292

```
# Choose the type of classifier
dtree_estimator = DecisionTreeClassifier(class_weight = {0: 0.17, 1: 0.83}, random_state = 1)

# Grid of parameters to choose from
parameters = {'max_depth': np.arange(2, 7),
              'criterion': ['gini', 'entropy'],
              'min_samples_leaf': [5, 10, 20, 25]
             }

# Type of scoring used to compare parameter combinations
scorer = metrics.make_scorer(recall_score, pos_label = 1)

# Run the grid search
gridCV = GridSearchCV(dtree_estimator, parameters, scoring = scorer, cv = 10)

# Fitting the grid search on the train data
gridCV = gridCV.fit(X_train, y_train)

# Set the classifier to the best combination of parameters
dtree_estimator = gridCV.best_estimator_

# Fit the best estimator to the data
dtree_estimator.fit(X_train, y_train)
```

✓ 1.9s

```
DecisionTreeClassifier(class_weight={0: 0.17, 1: 0.83}, criterion='entropy',
                       max_depth=5, min_samples_leaf=25, random_state=1)
```

- **Let's see how it performs on test set. Not overfitting but precision is extremely low.**

Decision Tree (Tuned)

- **Model's performance improved well on recall (from 0.64 to 0.73), but precision is low.**
- **It indicates that a large percentage of false positives will be created by the tuned model, i.e., the patient will be predicted to have covid while not having it. The hospital will spend time and resources on this.**

```
# Checking performance on the test dataset
y_test_pred_dt = ct.predict(X_test)

metrics_score(y_test, y_test_pred_dt)
```

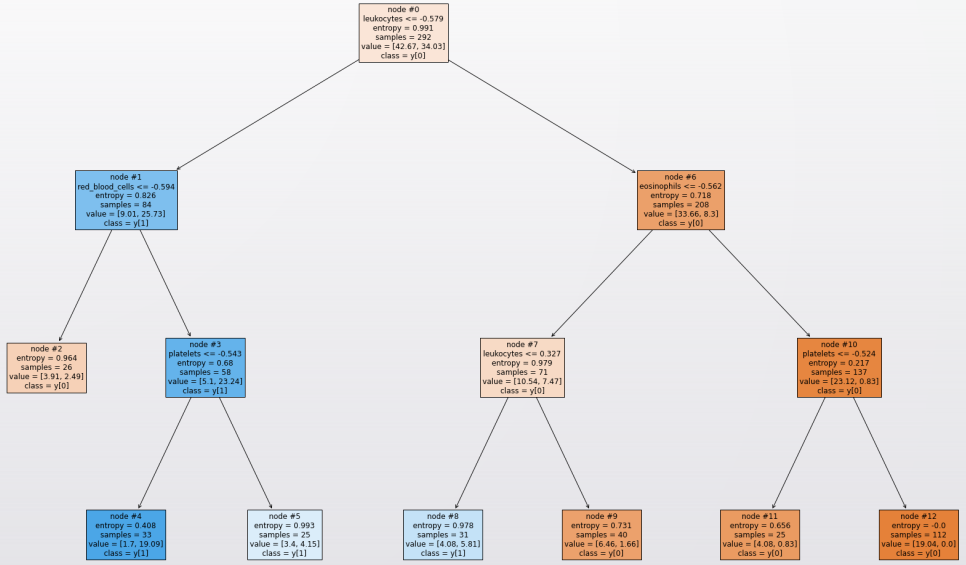
✓ 0.2s

	precision	recall	f1-score	support
0	0.94	0.79	0.86	63
1	0.38	0.73	0.50	11
accuracy			0.78	74
macro avg	0.66	0.76	0.68	74
weighted avg	0.86	0.78	0.81	74

Decision Tree (Tuned)

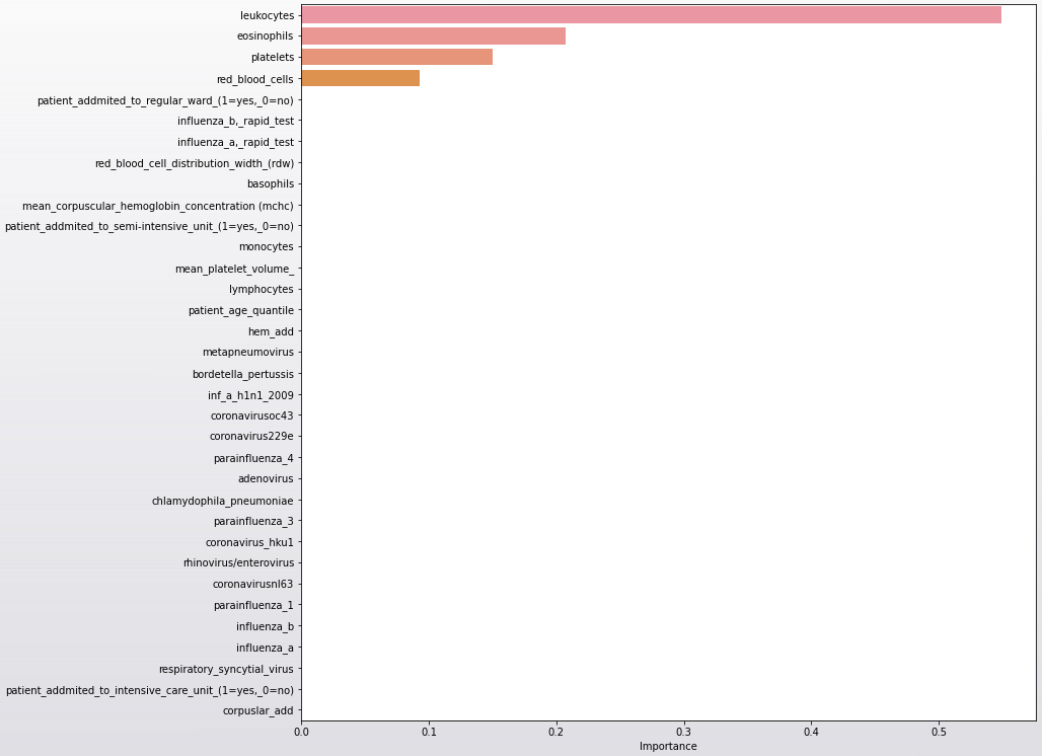
- **Leukocytes are at the top on the tree. We observe red blood cells and eosinophils at the second depth.**

- **We shall now look into the importance of the feature. This will provide useful insight into which features need to be prioritized.**



Feature importance

- **According to the Decision Tree, Leukocytes is the most important feature, followed by eosinophills, platelets and red_blood_cells.**
- **These features values can be associated with COVID's case, such as they may increase with COVID's presence. The features measurements can therefore be given top importance to identify the covid case.**



Random Forest

- **The random forest is returning 100% scores for all metrics on the training dataset.**
- **The train dataset has undoubtedly been overfit.**
- **This results in poor performance on the test dataset. Hospitals may not be able to predict positive cases with a 64% percentage, which is exceedingly risky for both the general public's health and the hospital's bottom line.**

```
# Checking performance on the training data
y_pred_train_rf = rf_estimator.predict(X_train)

metrics_score(y_train, y_pred_train_rf)
```

✓ 1.4s

	precision	recall	f1-score	support
0	1.00	1.00	1.00	251
1	1.00	1.00	1.00	41
accuracy			1.00	292
macro avg	1.00	1.00	1.00	292
weighted avg	1.00	1.00	1.00	292

```
# Checking performance on the testing data
y_pred_test_rf = rf_estimator.predict(X_test)
metrics_score(y_test, y_pred_test_rf)
```

✓ ✓ 0.2s

	precision	recall	f1-score	support
0	0.90	0.98	0.94	63
1	0.80	0.36	0.50	11
accuracy			0.89	74
macro avg	0.85	0.67	0.72	74
weighted avg	0.88	0.89	0.87	74

Random Forest (Tuned)

- **The tuned model is also slightly overfitting the training dataset, let's see how it performs on the test dataset.**

Random Forest (Tuned Model)

```
# Choose the type of classifier
rf_estimator_tuned = RandomForestClassifier(class_weight = {0: 0.17, 1: 0.83}, random_state = 1)

# Grid of parameters to choose from
params_rf = {
        "n_estimators": [25, 50, 75, 100, 150],
        "min_samples_leaf": np.arange(1, 4, 1),
        "max_features": [0.7, 0.9, 'auto'],
        "max_depth": [3, 4, 5, 'auto'],
}


# Type of scoring used to compare parameter combinations - recall score for class 1
scorer = metrics.make_scorer(recall_score, pos_label = 1)

# Run the grid search
grid_obj = GridSearchCV(rf_estimator_tuned, params_rf, scoring = scorer, cv = 5)

grid_obj = grid_obj.fit(X_train, y_train)

# Set the classifier to the best combination of parameters
rf_estimator_tuned = grid_obj.best_estimator_
```

```
# Checking performance on the training data
y_pred_train_rf_tuned = rf_estimator_tuned.predict(X_train)
metrics_score(y_train, y_pred_train_rf_tuned)
```

	precision	recall	f1-score	support
0	0.97	0.98	0.97	251
1	0.85	0.80	0.83	41
accuracy			0.95	292
macro avg	0.91	0.89	0.90	292
weighted avg	0.95	0.95	0.95	292

Random Forest (Tuned)

- **it shows a better performance on the test dataset compared to the model with no tuning.**
- **Recall increased from 0.36 to 0.64. Precision is 0.88, which is much better than decision tree.**

```
# Checking performance on the test data
y_pred_test_rf_tuned = rf_estimator_tuned.predict(X_test)

metrics_score(y_test, y_pred_test_rf_tuned)
```

✓ 0.3s

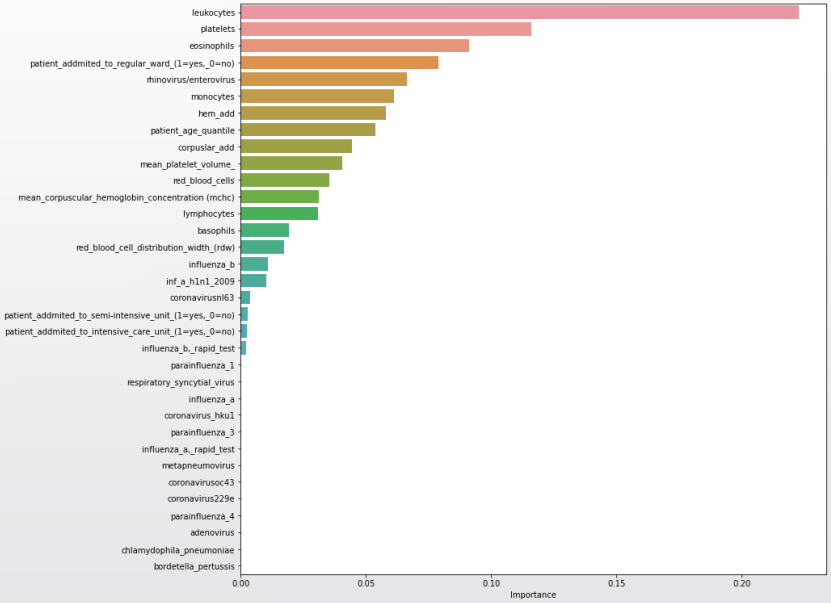
	precision	recall	f1-score	support
0	0.94	0.98	0.96	63
1	0.88	0.64	0.74	11
accuracy			0.93	74
macro avg	0.91	0.81	0.85	74
weighted avg	0.93	0.93	0.93	74

Feature importance

****Observations:****

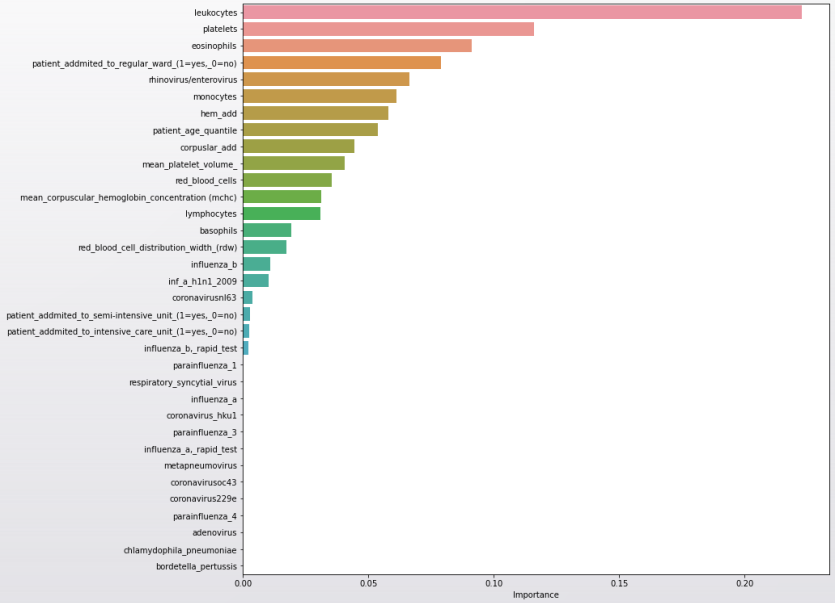
- **The Random Forest further verifies the results from the decision tree that the most important features are `Leukocytes`, `Plateles`, `Eosinophils`. However, it finds connection with other variables as well.**

- **Patient admitted to regular ward also seems to be significant component. It makes sense because it is very likely that people will be admitted to regular ward if they get covid.**

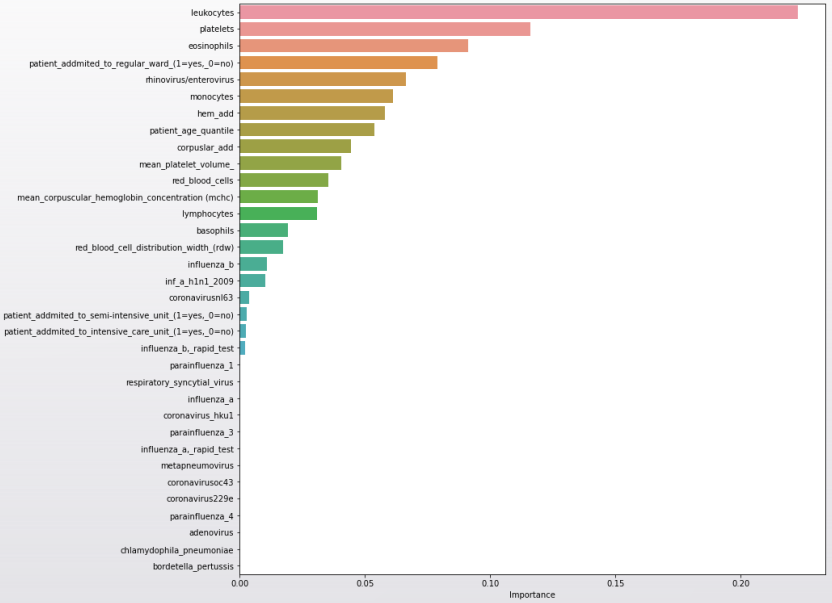


Feature importance

- **Observations:**
- rhinovirus/enterovirus seems to be associated with covid case because Rhinovirus/Enterovirus (RV/EV) was commonly found (83.3%) in co-infection with SARS-CoV-2 in adults.
- Reference:https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8402816/
- monocytes seems to be correlated with COVID's results because SARS-CoV-2 patients can show a predominant monocyte-derived macrophage infiltration in the severely damaged lungs
- Reference: https://www.dovepress.com/role-of monocytesmacrophages-in-covid-19-pathogenesis-implications-for-peer-reviewed-fulltext-article-IDR



Feature importance



- ****Observations:****
-

- hem_add, which is summation of hemoglobin and hematocrit, is also big component in the COVID's cases.

- **Some research shows that**
- **hemoglobin value was significantly lower in COVID-19 patients with severe disease**
- **Reference https://www.the-hospitalist.org/hospitalist/article/220824/coronavirus-updates/severe-covid-19-may-lower-hemoglobin-levels**

Results: Predictive Analysis (Classification)

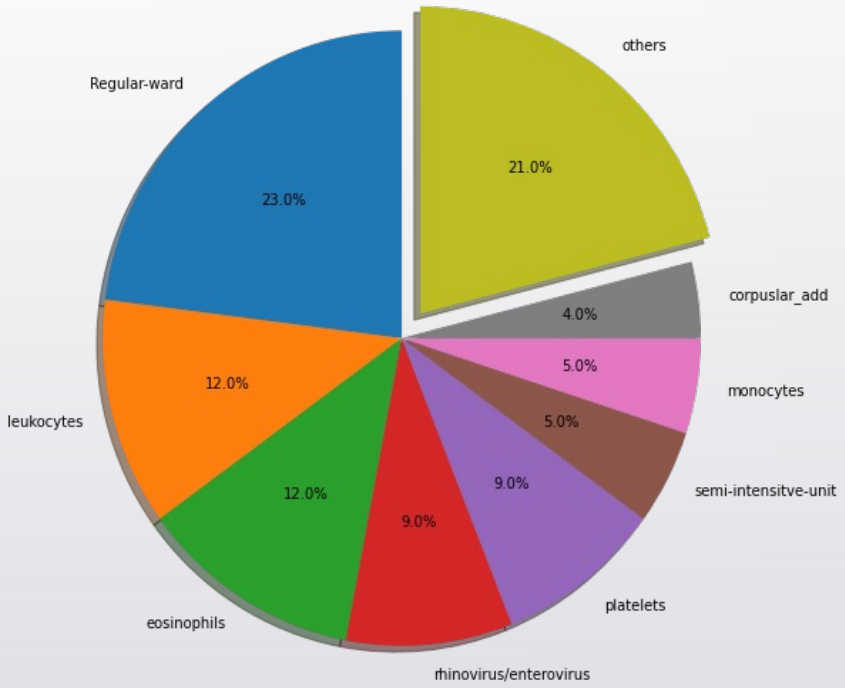
	Tr	Te	Tr	Te
	Recall		Precision	
SVM (RBF Kernel)	**0.83**	**0.82**	**0.87**	**0.82**
Random Forest	0.80	0.64	0.85	0.88
Logistic Regression	0.63	0.64	0.96	0.88

- **Imputation was applied and multicollinearity was removed using "variance inflation factor" from the statsmodels library.**
- **Precision-Recall Curve was used in SVM and Logistic Regression.**
- **Tuning with best set of parameters was applied to improve overall performance.**

Feature importance

- **Observations:**
- **Corpuslar add = mean_corpuscular_volume_(mcv)**
- **+ mean_corpuscular_hemoglobin_(mch)**

- **8 features cover 79 of total feature importance. Others include the rest of the features such as lymphocytes, basophils, red_blood_cells…**



Conclusions and Recommendations

- We have tried multiple models and identified the key factors involved with covid's positive and negative cases.

- Of all the models, logistic regression had the best performance. The results of train and test data, however, differ significantly. This raises some questions about the model's performance. Therefore, we don't recommend this model.

- Decision Tree precision is very low (0.31), although the recall good enough. The patient will be predicted to have covid while not having it. The hospital will spend time and resources on this. Also, the model's tuning showed that only 4 features significant. This might imply that the model fails to recognize the patterns linked to other variables.

- Random Forest has less recall, F1 Score and Accuracy than SVM, but precision is better (0.88). Nevertheless, we don't recommend this model because recall is lower than 0.7 with tuning.

Conclusions and Recommendations

- SVM with RBF kernel has good recall (0.82) among all the models with an F-1 score of 0.82, accuracy of 0.95 and precision of 0.82. **We recommend the deployment of this model among all other models because hospital will make right predictions as well as spend less time and budget with using this model.**

- We advise better data collection methods, **especially in Leukocytes, eosinophils, rhinovirus, plateles and monocytes,** because only a small number of features overall had missing values of less than 76 %. This caused us to train our model on sparse data and have overfitting issues with trainset frequently.