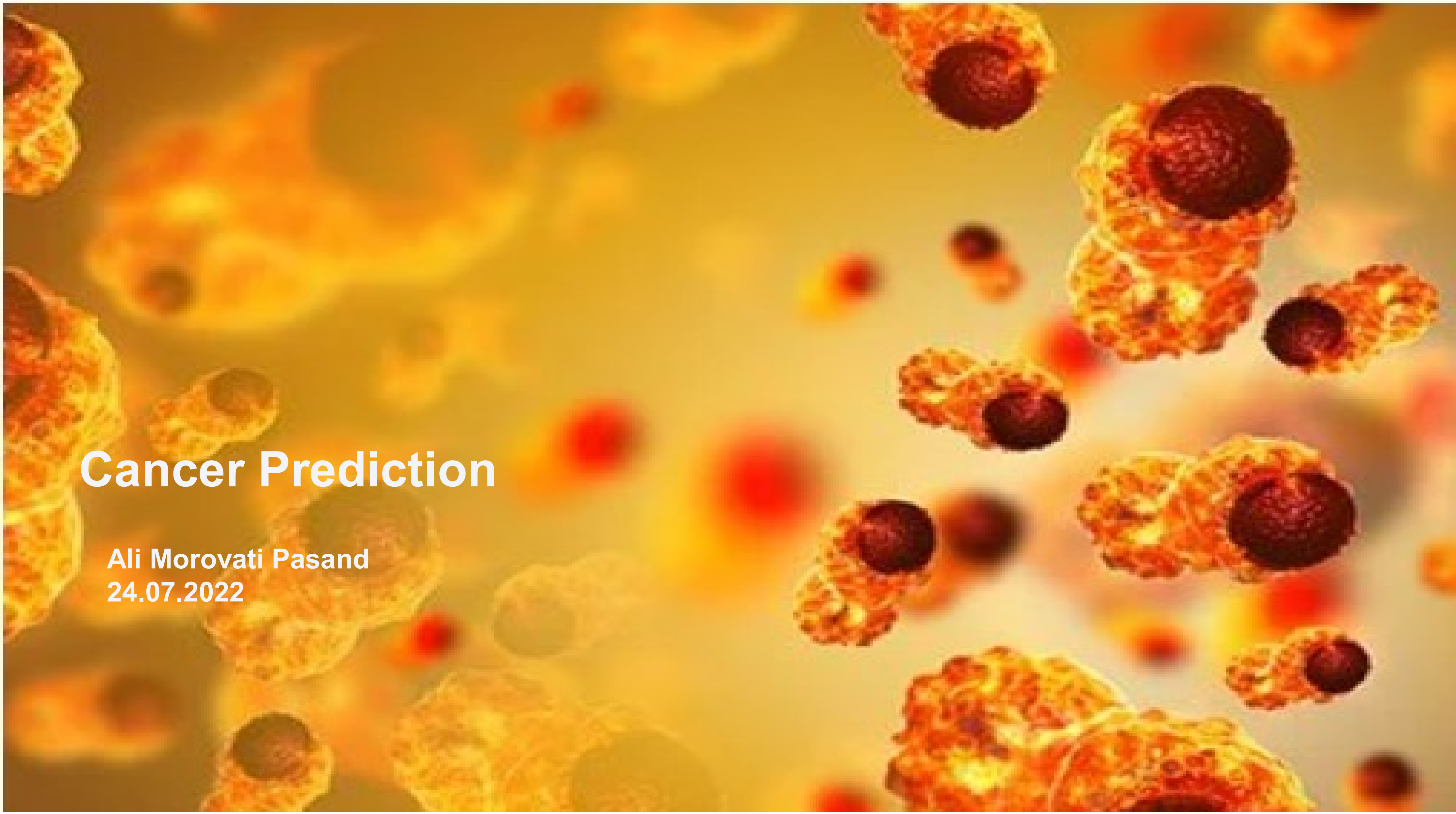


Cancer Prediction

Ali Morovati Pasand
24.07.2022

Outline

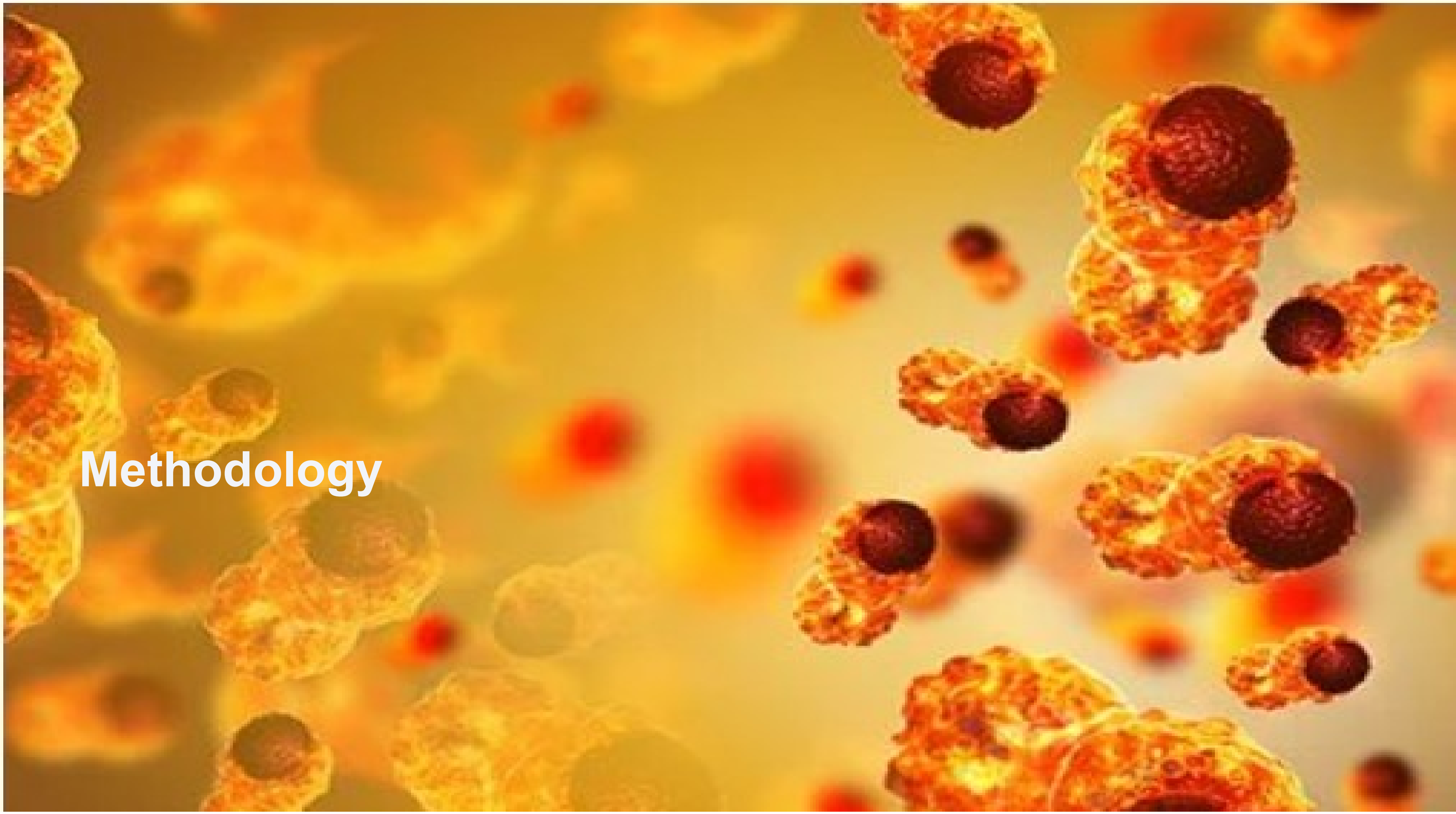
Introduction

- **Background of the project:**

  During the IBM course on Coursera we have done some exercises. One of them was a cancer prediction with SVM (Support Vector Machine) method. After completing the course, I used the same data-set and have decided to improve the prediction based on what I have learned to have a better predicting model.

- **Problems we want to find answers:**

  Classification of samples of tumor to find out whether it is benign or malignant based on the attributes of the tumor.



Methodology

Methodology

Executive Summary

- Perform data wrangling
- Perform interactive visual analytics using Plotly Dash
- Perform predictive analysis using classification models

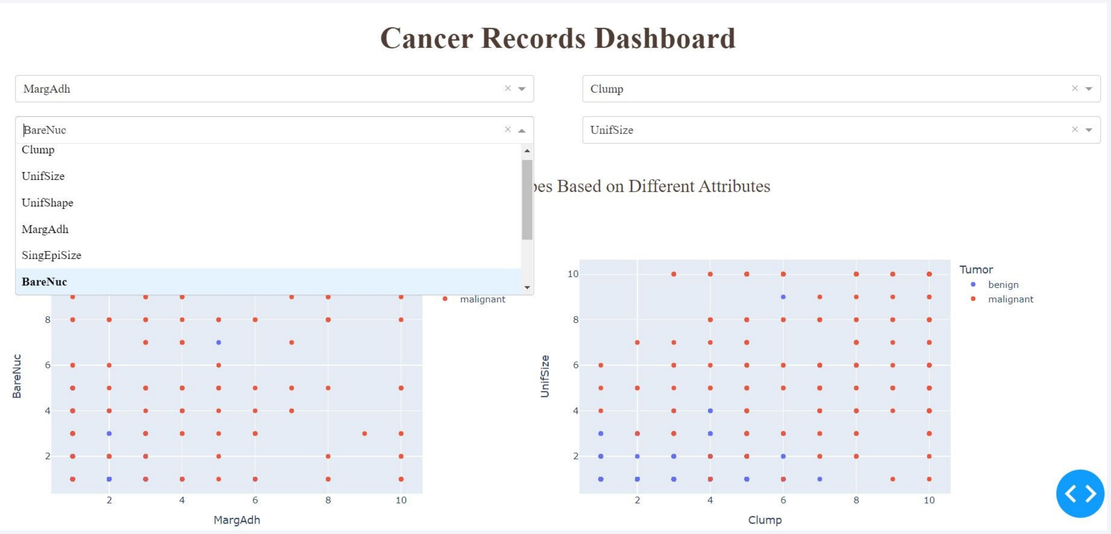
Data Wrangling

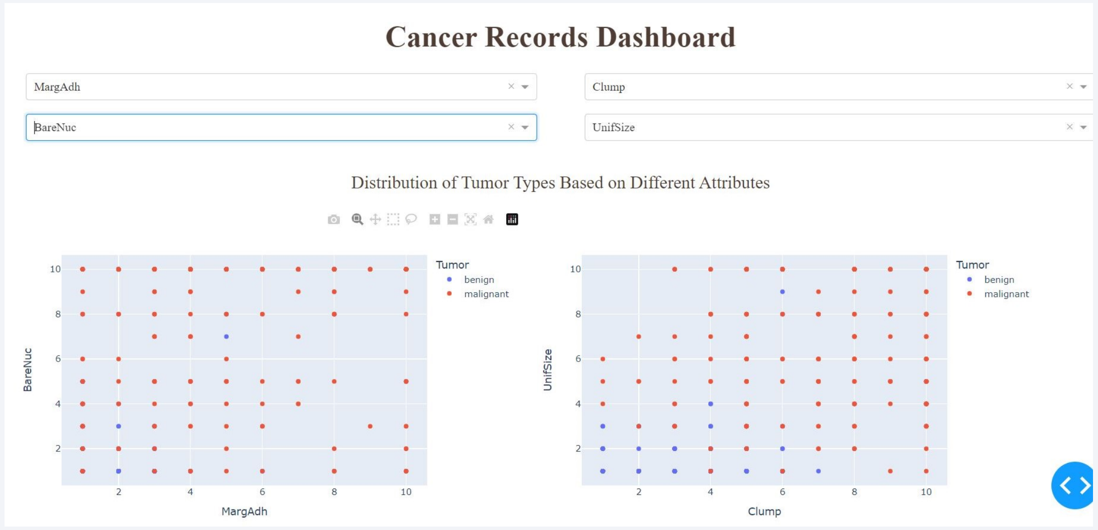
- Data wrangling in this project is as follows:
  - Creating a data frame from csv file
  - Get the shape of the data frame
  - Create a new column called Tumor to be used for plot legend
  - Get the data types of each column
  - Convert the non numeric values (numbers that are defined as an object) to numeric ones
  - Replace all the non digits (here “?”) with NaN
  - Remove the NaNs
  - Convert the values to integer type

EDA with Data Visualization

- Interactive visual analytics using Plotly Dash

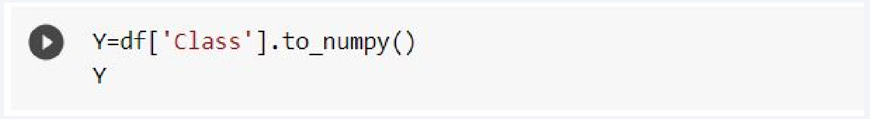
  The attributes are selected from drop-down list and scatter plot is plotted





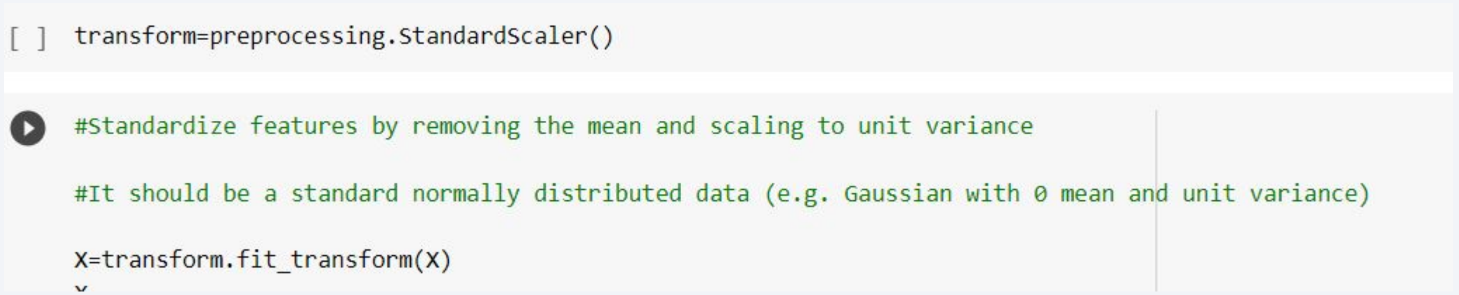
Predictive Analysis (Classification)

- For predictive analysis
  - The target values list is converted into a numpy array



```
Y=df['Class'].to_numpy()
Y
```

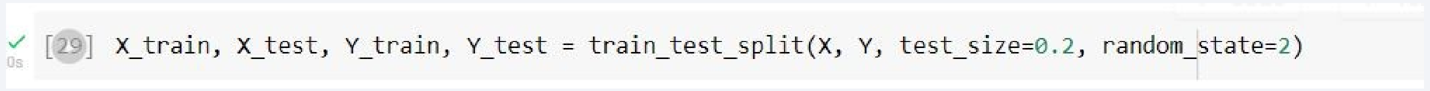
  - Selected features are converted to have 0 mean and unit variance



```
[ ]  transform=preprocessing.StandardScaler()
```

```
#Standardize features by removing the mean and scaling to unit variance

#It should be a standard normally distributed data (e.g. Gaussian with 0 mean and unit variance)

X=transform.fit_transform(X)
```

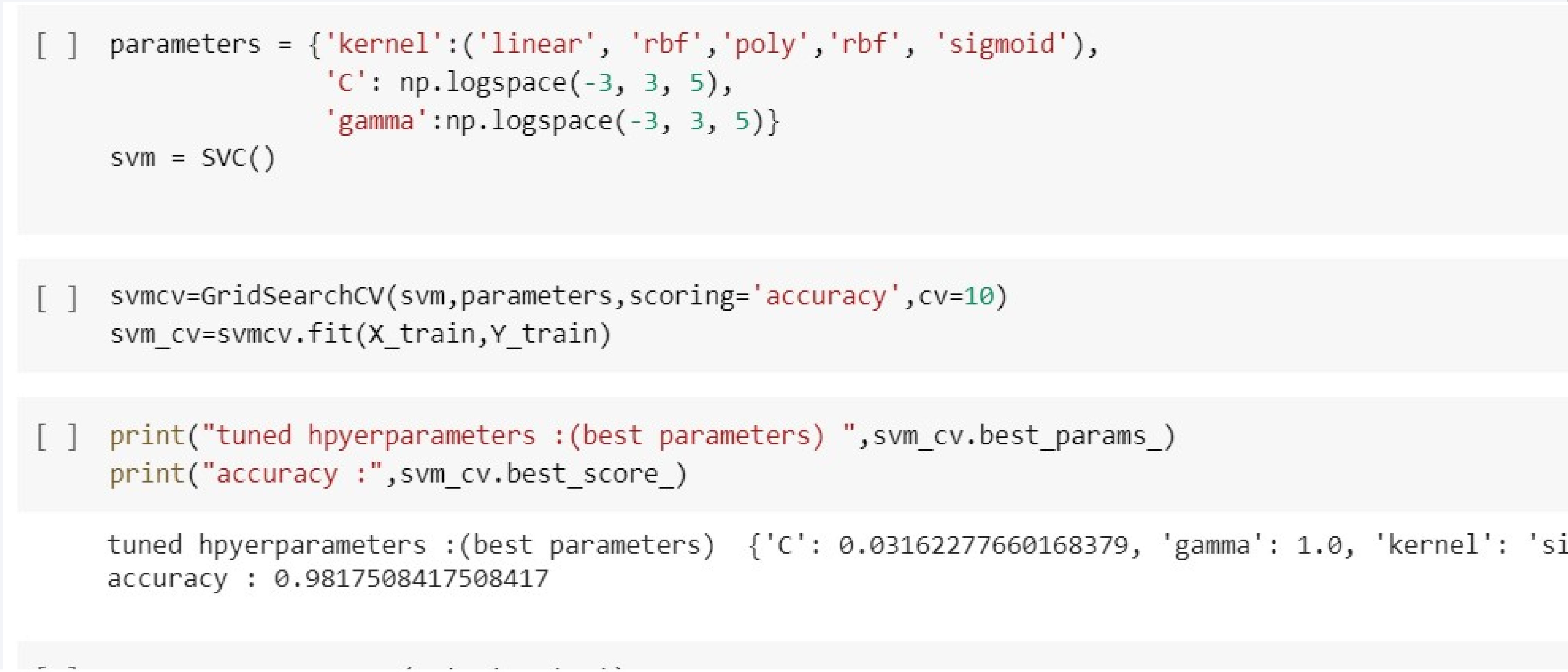
  - The data set is splitted into train and test sets



```
[29] X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
```

Predictive Analysis (Classification)

- Defining the parameters for the classification model (e.g. Support Vector Machine) and using the grid search method to find the best parameters



```
[ ]  parameters = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
                   'C': np.logspace(-3, 3, 5),
                   'gamma':np.logspace(-3, 3, 5)}
     svm = SVC()
```

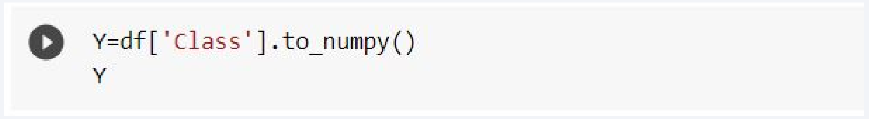
```
[ ]  svmcv=GridSearchCV(svm,parameters,scoring='accuracy',cv=10)
     svm_cv=svmcv.fit(X_train,Y_train)
```

```
[ ]  print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)
     print("accuracy :",svm_cv.best_score_)
```

```
     tuned hpyerparameters :(best parameters)  {'C': 0.03162277660168379, 'gamma': 1.0, 'kernel': 'si
     accuracy : 0.9817508417508417
```

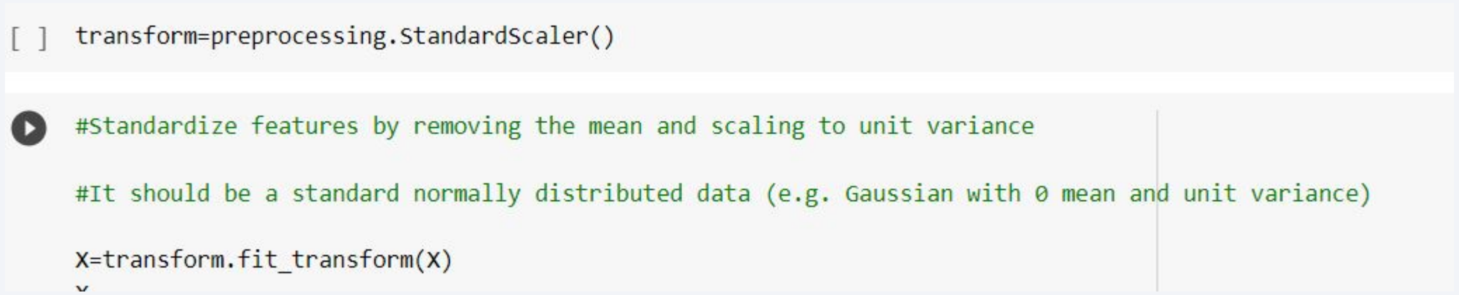
Predictive Analysis (Classification)

- For predictive analysis
  - The target values list is converted into a numpy array



```
Y=df['Class'].to_numpy()
Y
```

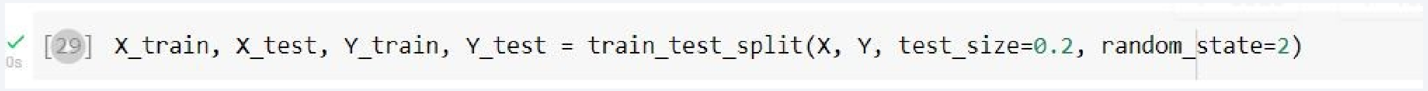
  - Selected features are converted to have 0 mean and unit variance



```
[ ]  transform=preprocessing.StandardScaler()
```

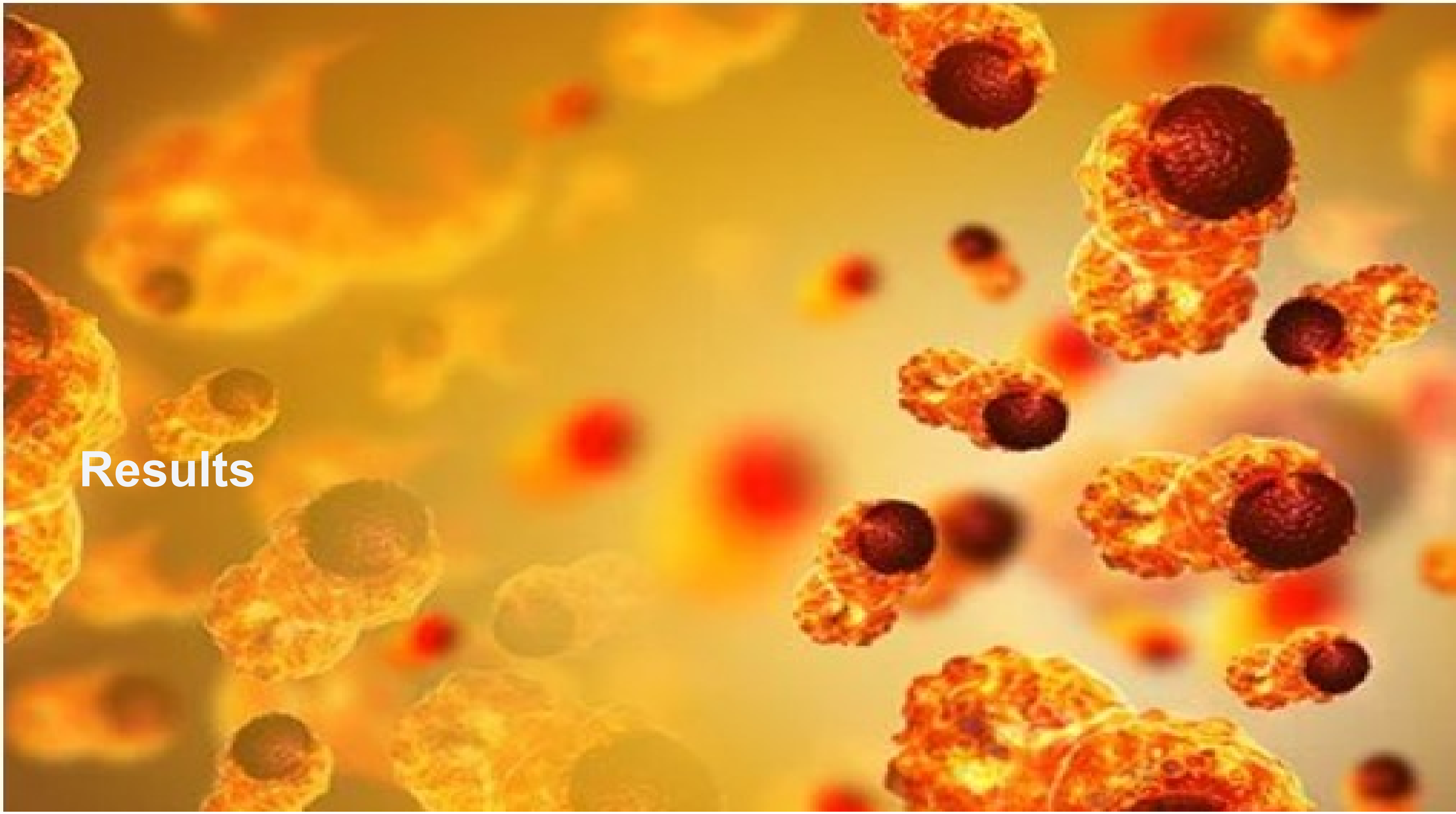
```
#Standardize features by removing the mean and scaling to unit variance

#It should be a standard normally distributed data (e.g. Gaussian with 0 mean and unit variance)

X=transform.fit_transform(X)
```

  - The data set is splitted into train and test sets

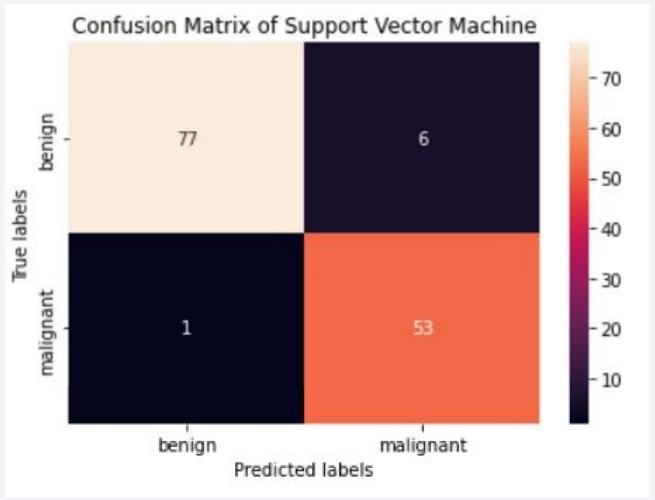


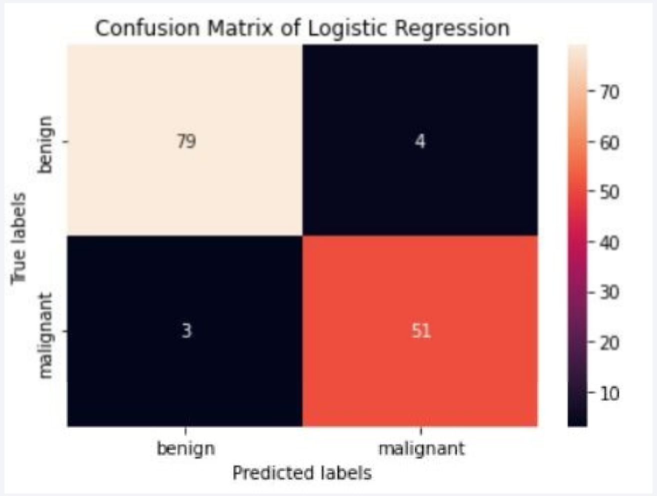
```
[29] X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
```

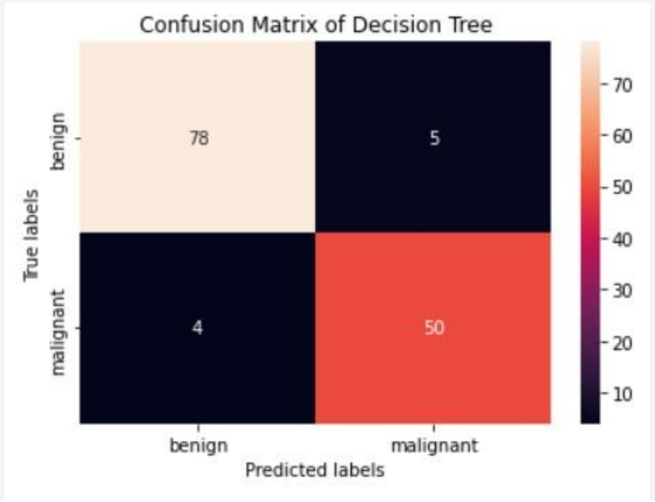
Results

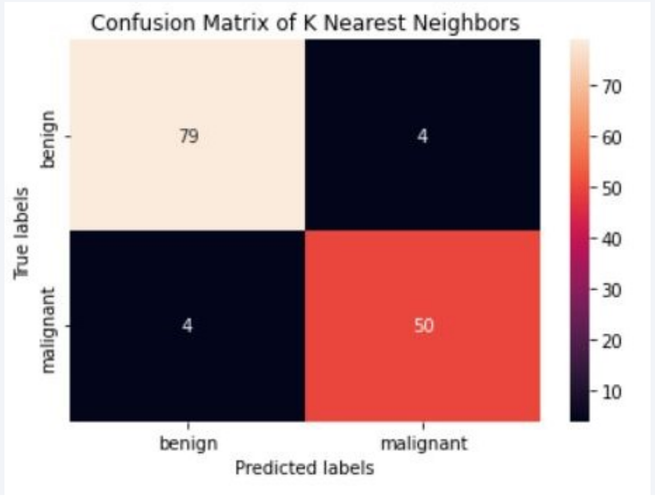


Results- Confusion Matrix







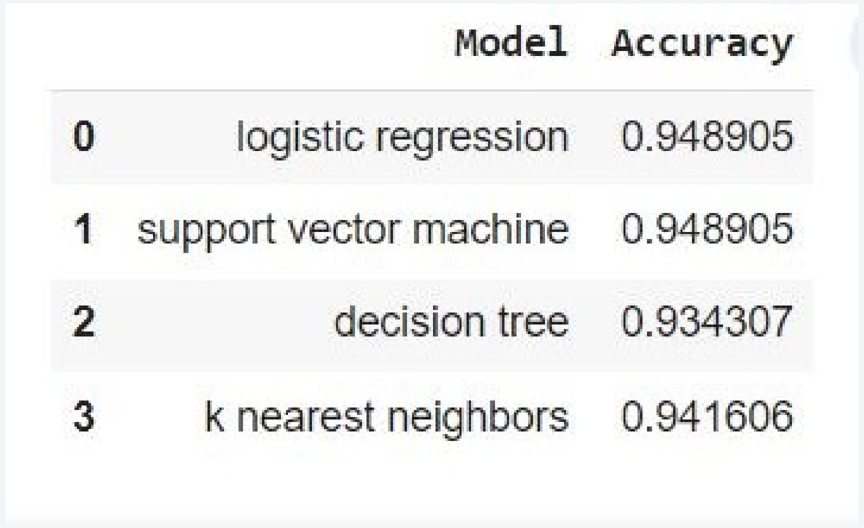


Results and Conclusion

- According to our predictive analyses, the best scores belong to Logistic Regression (LR) and Support Vector Machine(SVM). The best parameters for these two models are:
  - SVM

    C=0.031, gamma=1.0, Kernel=Sigmoid
  - LR

    C= 1, penalty= l2, solver= lbfgs



	Model	Accuracy
0	logistic regression	0.948905
1	support vector machine	0.948905
2	decision tree	0.934307
3	k nearest neighbors	0.941606

Appendix

- Github link of the notebook file:

https://github.com/Amorovati/Cancer_Prediction