Machine Learning Model Prediction

Data Preparation

Required Specific Libraries

```
# Import necessary libraries
import numpy as np

import pandas as pd

import matplotlib.pyplot as matplt
from matplotlib import rcParams
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

# Import data preprocessing libraries and models
from sklearn.preprocessing import StandardScaler, LabelEncoder, OrdinalEncoder, OneHotEncoder
from sklearn.compose import make_column_transformer
from imblearn.over_sampling import SMOTE

# Import model evaluation and training utilities
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,ConfusionMatrixDisplay, accuracy_score, confusion_matrix, precision_score, recall_score

# Import machine learning models
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression


# Mount Google Drive to access datasets
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

Import dataset

```
# Load the dataset into a DataFrame
dataframe = pd.read_csv('/content/drive/My Drive/Colab Notebooks/Exam/LA4PSchools.csv')


dataframe.head()
```

	StudentID	Gender	Year_02	Kinder_Age	Disability	NCCD-Funded	01.SES	02.SES	NumSibling	SiblingOrder	...	HRSIW-01-SOY	Counting-01	Coun
0	SN35433053	Male	2020	5.5	Disability_Non-disable	0	104	104	3	3	...	49	4	
1	SN71277215	Female	2018	5.8	Disability_Non-disable	0	112	112	2	2	...	37	2	
2	SN40883127	Male	2021	5.9	Disability_Non-disable	0	120	109	2	2	...	30	2	
3	SN93063777	Male	2021	5.7	Disability_Non-disable	0	95	93	2	1	...	30	2	
4	SN84195329	Male	2021	5.8	Disability_Non-disable	0	98	98	1	1	...	32	2	

5 rows × 34 columns

```
dataframe.tail()
```

	StudentID	Gender	Year_02	Kinder_Age	Disability	NCCD-Funded	01.SES	02.SES	NumSibling	SiblingOrder	...	HRSIW-01-SOY	Counting-01
1995	SN72182891	Female	2016	4.9	Disability_Cognitive	0	114	114	2	1	...	32	
1996	SN69853705	Female	2021	4.6	Disability_Non-disable	0	95	93	1	1	...	45	
1997	SN64915269	Male	2016	5.7	Disability_Non-disable	0	101	101	3	2	...	33	
1998	SN70943748	Female	2021	5.4	Disability_Non-disable	0	117	108	3	1	...	35	
1999	SN70826435	Male	2017	5.6	Disability_Cognitive	0	95	95	2	2	...	19	

5 rows × 34 columns

```
print('No of tuples: ', dataframe.shape[0],'| No of features: ',dataframe.shape[1])
```

```
No of tuples:  2000 | No of features:  34
```

```
dataframe.describe()
```

	Year_02	Kinder_Age	NCCD-Funded	01.SES	02.SES	NumSibling	SiblingOrder	NumAbvYear9	NumAbvDiploma	NumProf	.
count	2000.000000	2000.000000	2000.000000	2000.0000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	
mean	2018.640000	5.276400	0.089000	102.9415	102.117500	2.356500	1.748500	1.564000	0.886500	0.766500	
std	1.664568	0.348129	0.284815	9.3859	9.150167	0.993932	0.865229	0.725374	0.837836	0.811977	
min	2016.000000	4.500000	0.000000	78.0000	78.000000	1.000000	1.000000	0.000000	0.000000	0.000000	
25%	2017.000000	5.000000	0.000000	95.0000	95.000000	2.000000	1.000000	1.000000	0.000000	0.000000	
50%	2018.000000	5.300000	0.000000	101.0000	101.000000	2.000000	2.000000	2.000000	1.000000	1.000000	
75%	2020.000000	5.500000	0.000000	113.0000	109.000000	3.000000	2.000000	2.000000	2.000000	1.000000	
max	2021.000000	6.500000	1.000000	120.0000	120.000000	7.000000	6.000000	3.000000	2.000000	2.000000	

8 rows × 30 columns

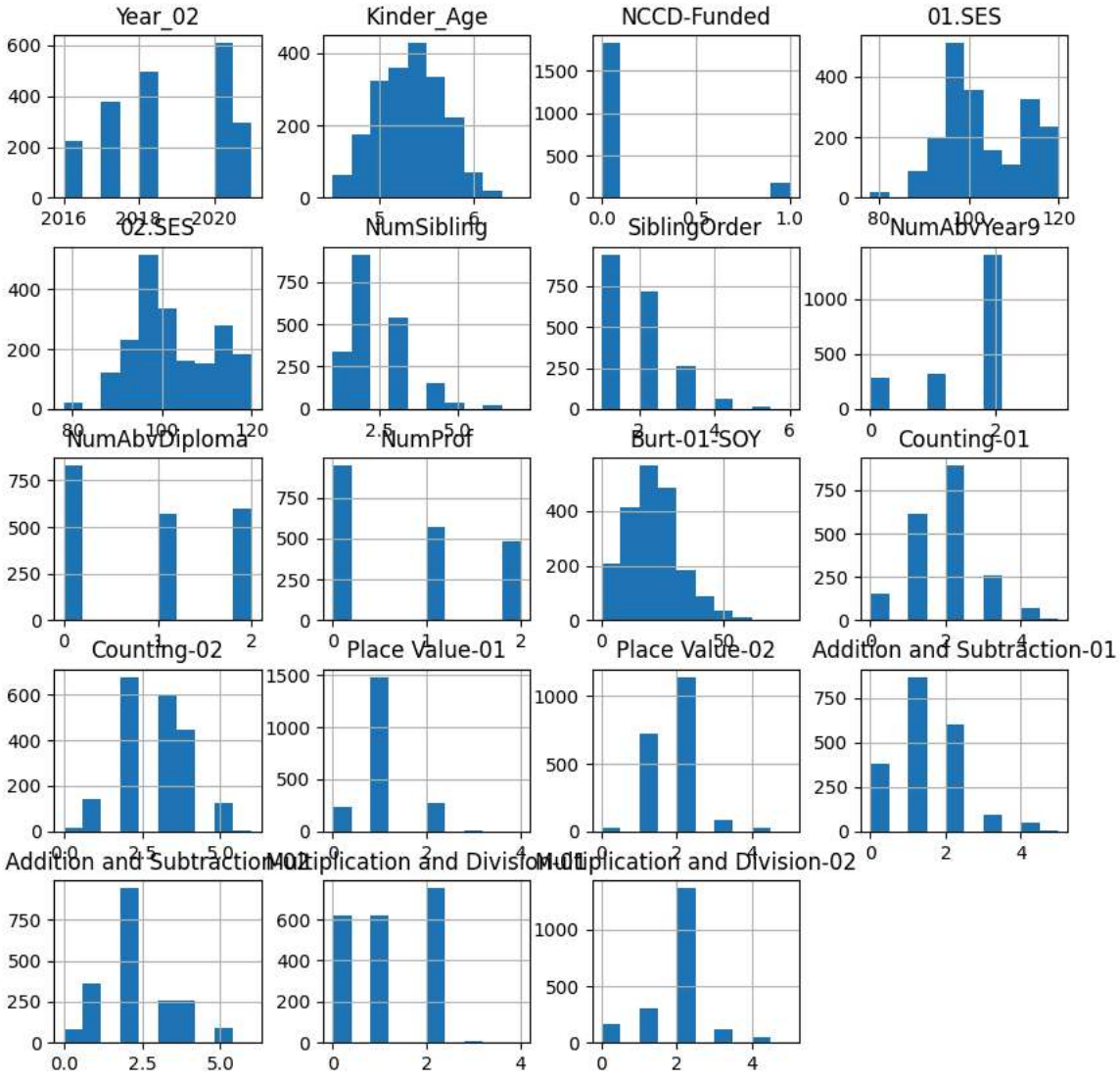
```
# add targat variable as a boolean
dataframe['Year3_Writing_At_Risk'] = dataframe['Year3_Writing_At_Risk'].map({False: 0, True: 1})
```

```
dataframe.describe(include=["O"])
```

	StudentID	Gender	Disability
count	2000	2000	2000
unique	2000	2	5
top	SN35433053	Male	Disability_Non-disable
freq	1	1018	1381

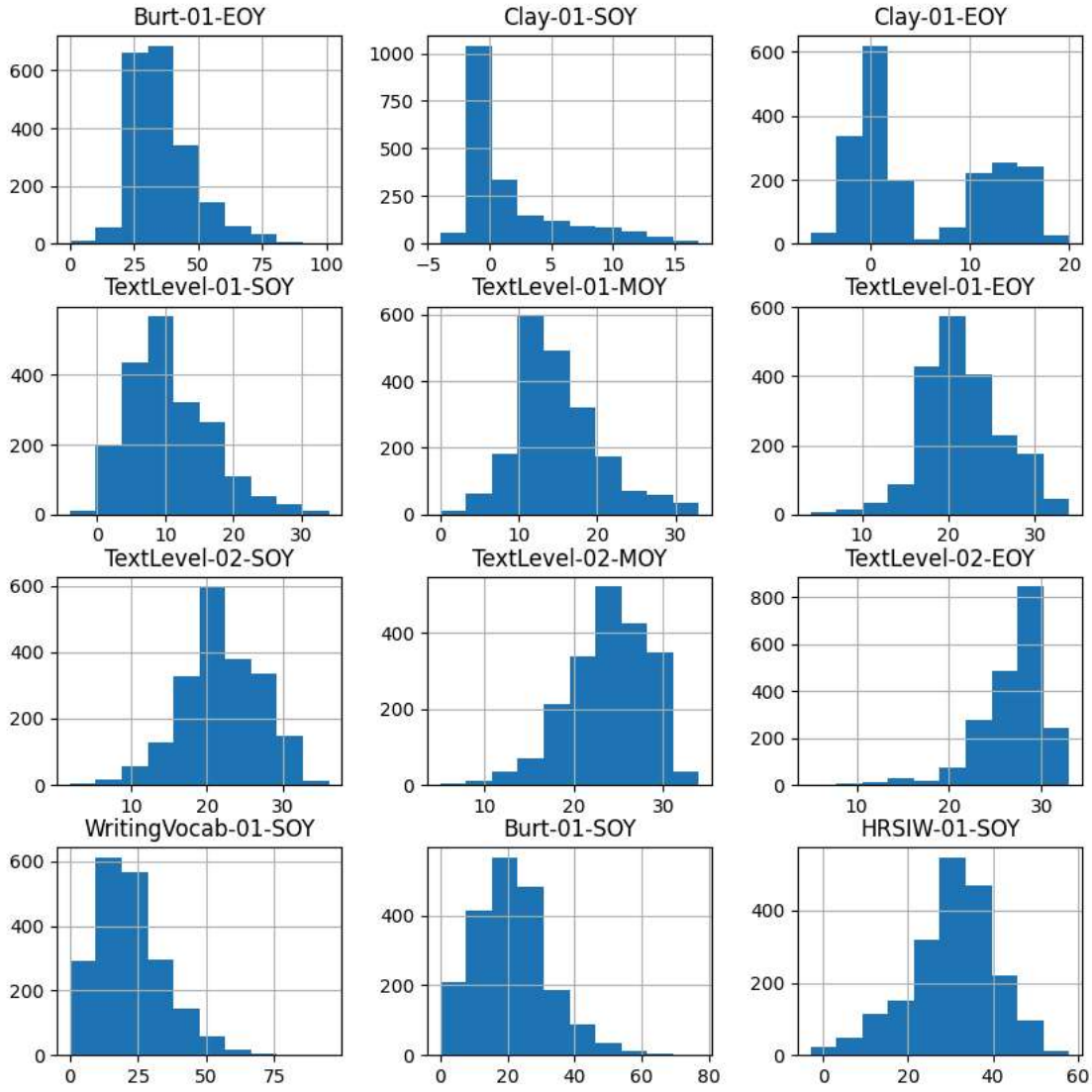
```
rcParams['figure.figsize'] = 10, 10
dataframe[['Year_02', 'Kinder_Age', 'NCCD-Funded', '01.SES', "02.SES", "NumSibling", "SiblingOrder", "NumAbvYear9", "NumAbvDiploma", "NumPrc
```

```
array([[<Axes: title={'center': 'Year_02'}>,
        <Axes: title={'center': 'Kinder_Age'}>,
        <Axes: title={'center': 'NCCD-Funded'}>,
        <Axes: title={'center': '01.SES'}>],
       [<Axes: title={'center': '02.SES'}>,
        <Axes: title={'center': 'NumSibling'}>,
        <Axes: title={'center': 'SiblingOrder'}>,
        <Axes: title={'center': 'NumAbvYear9'}>],
       [<Axes: title={'center': 'NumAbvDiploma'}>,
        <Axes: title={'center': 'NumProf'}>,
        <Axes: title={'center': 'Burt-01-SOY'}>,
        <Axes: title={'center': 'Counting-01'}>],
       [<Axes: title={'center': 'Counting-02'}>,
        <Axes: title={'center': 'Place Value-01'}>,
        <Axes: title={'center': 'Place Value-02'}>,
        <Axes: title={'center': 'Addition and Subtraction-01'}>],
       [<Axes: title={'center': 'Addition and Subtraction-02'}>,
        <Axes: title={'center': 'Multiplication and Division-01'}>,
        <Axes: title={'center': 'Multiplication and Division-02'}>,
        <Axes: >]], dtype=object)
```




```
rcParams['figure.figsize'] = 10, 10
dataframe[['Burt-01-EOY', 'Clay-01-SOY', 'Clay-01-EOY', 'TextLevel-01-SOY', "TextLevel-01-MOY", "TextLevel-01-EOY", "TextLevel-02-SOY", "Tex
```

```
array([[<Axes: title={'center': 'Burt-01-EOY'}>,
        <Axes: title={'center': 'Clay-01-SOY'}>,
        <Axes: title={'center': 'Clay-01-EOY'}>],
       [<Axes: title={'center': 'TextLevel-01-SOY'}>,
        <Axes: title={'center': 'TextLevel-01-MOY'}>,
        <Axes: title={'center': 'TextLevel-01-EOY'}>],
       [<Axes: title={'center': 'TextLevel-02-SOY'}>,
        <Axes: title={'center': 'TextLevel-02-MOY'}>,
        <Axes: title={'center': 'TextLevel-02-EOY'}>],
       [<Axes: title={'center': 'WritingVocab-01-SOY'}>,
        <Axes: title={'center': 'Burt-01-SOY'}>,
        <Axes: title={'center': 'HRSIW-01-SOY'}>]], dtype=object)
```

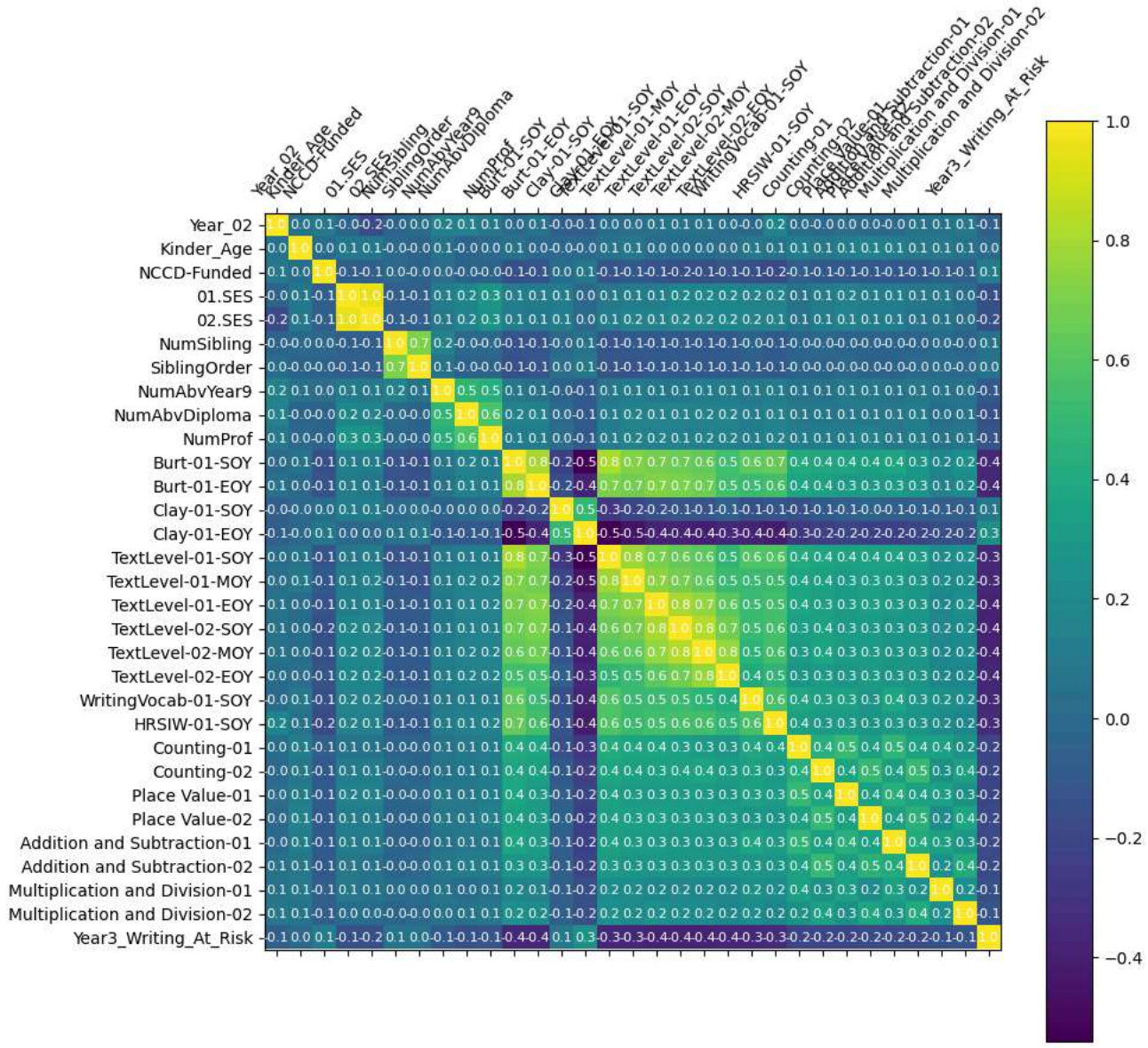



```
print(dataframe.dtypes)
```

```
StudentID                 object
Gender                    object
Year_02                    int64
Kinder_Age               float64
Disability                object
NCCD-Funded                int64
01.SES                     int64
02.SES                     int64
NumSibling                 int64
SiblingOrder               int64
NumAbvYear9                int64
NumAbvDiploma              int64
NumProf                    int64
Burt-01-SOY                int64
Burt-01-EOY                int64
Clay-01-SOY                int64
Clay-01-EOY                int64
TextLevel-01-SOY           int64
TextLevel-01-MOY           int64
```

```
TextLevel-01-EOY                   int64
TextLevel-02-SOY                   int64
TextLevel-02-MOY                   int64
TextLevel-02-EOY                   int64
WritingVocab-01-SOY                int64
HRSIW-01-SOY                       int64
Counting-01                        int64
Counting-02                        int64
Place Value-01                     int64
Place Value-02                     int64
Addition and Subtraction-01        int64
Addition and Subtraction-02        int64
Multiplication and Division-01     int64
Multiplication and Division-02     int64
Year3_Writing_At_Risk              int64
dtype: object
```

```
numeric_df = dataframe.select_dtypes(exclude=['object'])


# Create correlation matrix plot for numeric features
fig = matplt.figure(figsize=(10, 10))

matplt.matshow(numeric_df.corr(),fignum=fig.number) # Correlation heatmap
matplt.colorbar()
matplt.xticks(np.arange(len(numeric_df.corr().columns)), numeric_df.corr().columns.values, rotation = 50)
matplt.yticks(np.arange(len(numeric_df.corr().columns)), numeric_df.corr().columns.values)

# Show correlation values inside the matrix
for (i, j), corr in np.ndenumerate(numeric_df.corr()):
    matplt.text(j, i, '{:0.1f}'.format(corr), ha='center', va='center', color='white', fontsize=8)
```



Preprocess the data

Dropping unnecessary data

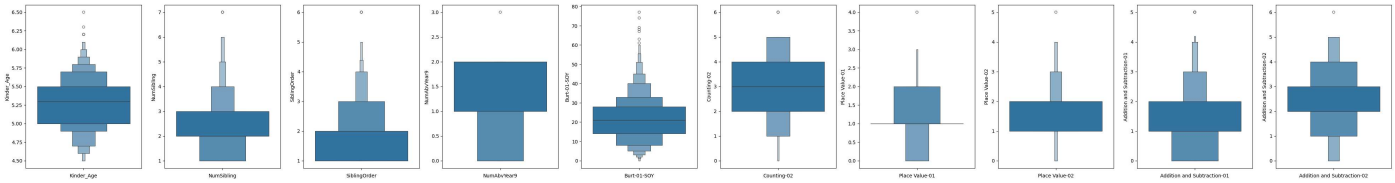
```
# Drop 'StudentID' column since it's not useful for the model
dataframe.drop(["StudentID"],axis=1,inplace=True)
```

```
dataframe.shape
```

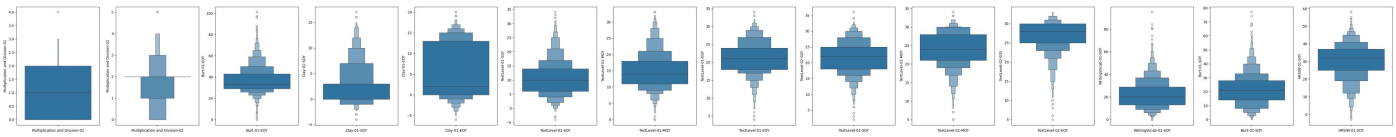
```
(2000, 33)
```

Outlier Removals

```
# Boxenplot visualization for selected columns to identify outliers
colomns = ['Kinder_Age', "NumSibling", "SiblingOrder", "NumAbvYear9", "Burt-01-SOY", "Counting-02", "Place Value-01", "Place Value-02", "Adc

figure, ax = matplt.subplots(ncols=len(colomns), figsize=(5 * len(colomns),6), sharex = True)

for r in range(len(colomns)):
  (sns.boxenplot(y=dataframe[colomns[r]],data=dataframe, showfliers=True,ax=ax[r])).set(xlabel=colomns[r])
```



```
# Boxenplot visualization for selected columns to identify outliers
colomns = ["Multiplication and Division-01", "Multiplication and Division-02",'Burt-01-EOY', 'Clay-01-SOY', 'Clay-01-EOY', 'TextLevel-01-SOY

figure, ax = matplt.subplots(ncols=len(colomns), figsize=(5 * len(colomns),6), sharex = True)

for r in range(len(colomns)):
  (sns.boxenplot(y=dataframe[colomns[r]],data=dataframe, showfliers=True,ax=ax[r])).set(xlabel=colomns[r])
```



For Encoding Categorical data

```
dataframe.describe(include=['object'])
```

	Gender	Disability
count	2000	2000
unique	2	5
top	Male	Disability_Non-disable
freq	1018	1381

```
print(dataframe.dtypes)
```

```
Gender                              object
Year_02                              int64
Kinder_Age                         float64
Disability                          object
NCCD-Funded                          int64
01.SES                               int64
02.SES                               int64
NumSibling                           int64
SiblingOrder                         int64
NumAbvYear9                          int64
NumAbvDiploma                        int64
NumProf                              int64
Burt-01-SOY                          int64
Burt-01-EOY                          int64
Clay-01-SOY                          int64
Clay-01-EOY                          int64
TextLevel-01-SOY                     int64
TextLevel-01-MOY                     int64
TextLevel-01-EOY                     int64
TextLevel-02-SOY                     int64
TextLevel-02-MOY                     int64
TextLevel-02-EOY                     int64
WritingVocab-01-SOY                  int64
HRSIW-01-SOY                         int64
Counting-01                          int64
Counting-02                          int64
Place Value-01                       int64
Place Value-02                       int64
Addition and Subtraction-01          int64
Addition and Subtraction-02          int64
Multiplication and Division-01       int64
Multiplication and Division-02       int64
```

```
Year3_Writing_At_Risk               int64
dtype: object
```

```
for d in dataframe.select_dtypes(include = 'object'):
    print(dataframe[d].value_counts())
    print("")
```

```
Gender
Male      1018
Female     982
Name: count, dtype: int64

Disability
Disability_Non-disable        1381
Disability_Cognitive           469
Disability_SocialEmotional      73
Disability_Physical             67
Disability_Sensory              10
Name: count, dtype: int64
```

```
# Data transformation: replace categorical values for 'Gender'
change_values = {
  'Gender': {
    'Male': 0,
    'Female': 1
  }
}

dataframe.replace(change_values, inplace=True)
```

```
dataframe.head()
```

	Gender	Year_02	Kinder_Age	Disability	NCCD-Funded	01.SES	02.SES	NumSibling	SiblingOrder	NumAbvYear9	...	HRSIW-01-SOY	Counting-01	Coun
0	0	2020	5.5	Disability_Non-disable	0	104	104	3	3	2	...	49	4	
1	1	2018	5.8	Disability_Non-disable	0	112	112	2	2	2	...	37	2	
2	0	2021	5.9	Disability_Non-disable	0	120	109	2	2	2	...	30	2	
3	0	2021	5.7	Disability_Non-disable	0	95	93	2	1	0	...	30	2	
4	0	2021	5.8	Disability_Non-disable	0	98	98	1	1	2	...	32	2	

5 rows × 33 columns

```
print(dataframe.dtypes)
```

```
Gender                          int64
Year_02                         int64
Kinder_Age                    float64
Disability                     object
NCCD-Funded                     int64
01.SES                          int64
02.SES                          int64
NumSibling                      int64
SiblingOrder                    int64
NumAbvYear9                     int64
NumAbvDiploma                   int64
NumProf                         int64
Burt-01-SOY                     int64
Burt-01-EOY                     int64
Clay-01-SOY                     int64
Clay-01-EOY                     int64
TextLevel-01-SOY                int64
TextLevel-01-MOY                int64
TextLevel-01-EOY                int64
TextLevel-02-SOY                int64
TextLevel-02-MOY                int64
TextLevel-02-EOY                int64
WritingVocab-01-SOY             int64
```

```
HRSIW-01-SOY                      int64
Counting-01                       int64
Counting-02                       int64
Place Value-01                    int64
Place Value-02                    int64
Addition and Subtraction-01       int64
Addition and Subtraction-02       int64
Multiplication and Division-01    int64
Multiplication and Division-02    int64
Year3_Writing_At_Risk             int64
dtype: object
```

```
dataframe_copy = dataframe


# One-Hot Encode 'Disability' feature and merge with the original dataframe
disability_transformer = make_column_transformer(
  (OneHotEncoder(), ['Disability']),
  remainder='passthrough',
  verbose_feature_names_out=False
)

dataframe = dataframe_copy

disability_transformed = disability_transformer.fit_transform(dataframe)
dataframe = pd.DataFrame(disability_transformed, columns=disability_transformer.get_feature_names_out())
dataframe.head(20)
```

	Disability_Disability_Cognitive	Disability_Disability_Non-disable	Disability_Disability_Physical	Disability_Disability_Sensory	Disabili
0	0.0	1.0	0.0	0.0	
1	0.0	1.0	0.0	0.0	
2	0.0	1.0	0.0	0.0	
3	0.0	1.0	0.0	0.0	
4	0.0	1.0	0.0	0.0	
5	0.0	1.0	0.0	0.0	
6	0.0	1.0	0.0	0.0	
7	0.0	1.0	0.0	0.0	
8	0.0	1.0	0.0	0.0	
9	0.0	1.0	0.0	0.0	
10	0.0	1.0	0.0	0.0	
11	0.0	1.0	0.0	0.0	
12	0.0	1.0	0.0	0.0	
13	0.0	1.0	0.0	0.0	
14	0.0	1.0	0.0	0.0	
15	0.0	0.0	1.0	0.0	
16	1.0	0.0	0.0	0.0	
17	1.0	0.0	0.0	0.0	
18	0.0	1.0	0.0	0.0	
19	1.0	0.0	0.0	0.0	

20 rows × 37 columns

```
dataframe.shape
```

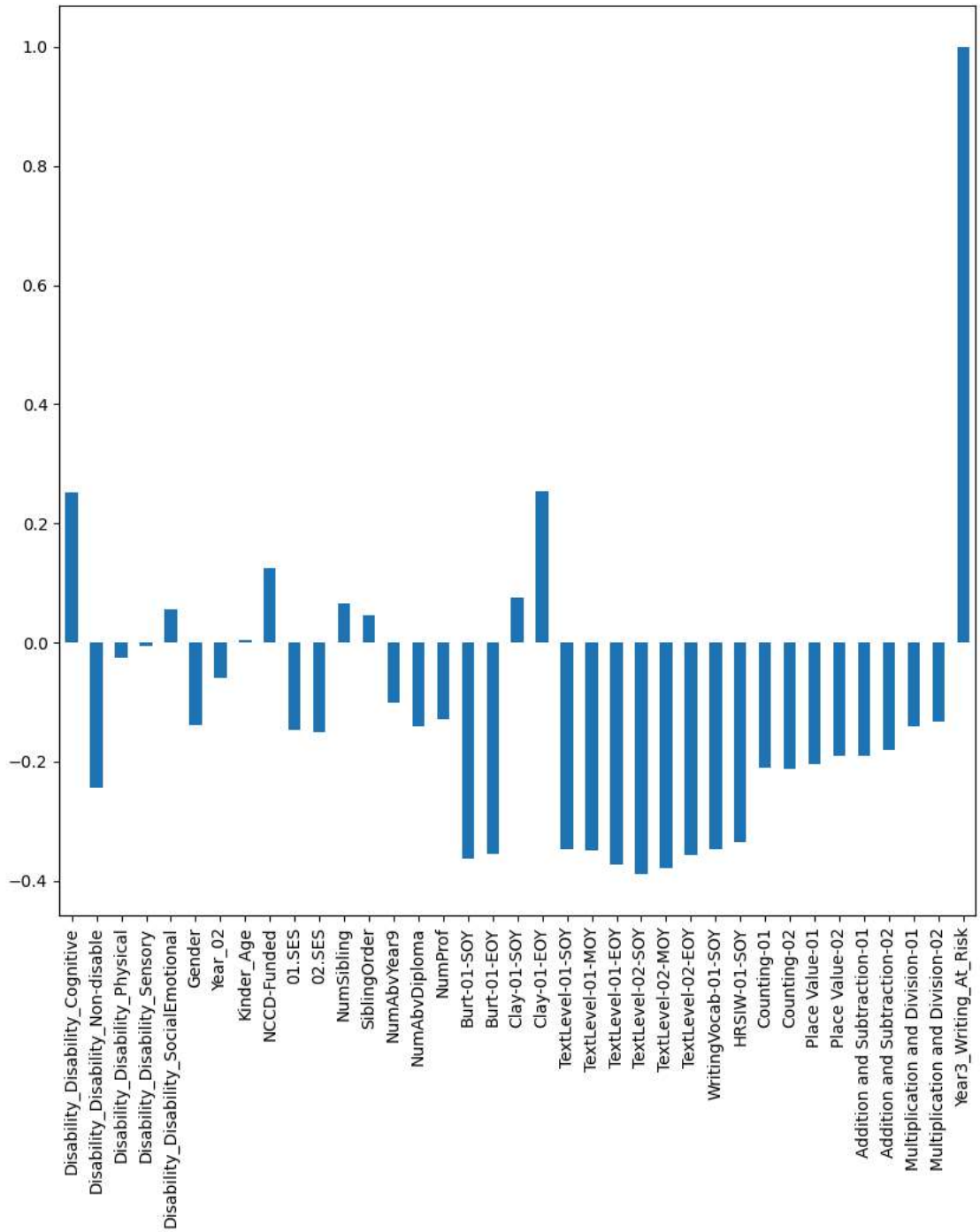
```
(2000, 37)
```

```
print(dataframe.dtypes)
```

```
Disability_Disability_Cognitive         float64
Disability_Disability_Non-disable       float64
Disability_Disability_Physical          float64
```

```
Disability_Disability_Sensory            float64
Disability_Disability_SocialEmotional    float64
Gender                                   float64
Year_02                                  float64
Kinder_Age                               float64
NCCD-Funded                              float64
01.SES                                   float64
02.SES                                   float64
NumSibling                               float64
SiblingOrder                             float64
NumAbvYear9                              float64
NumAbvDiploma                            float64
NumProf                                  float64
Burt-01-SOY                              float64
Burt-01-EOY                              float64
Clay-01-SOY                              float64
Clay-01-EOY                              float64
TextLevel-01-SOY                         float64
TextLevel-01-MOY                         float64
TextLevel-01-EOY                         float64
TextLevel-02-SOY                         float64
TextLevel-02-MOY                         float64
TextLevel-02-EOY                         float64
WritingVocab-01-SOY                      float64
HRSIW-01-SOY                             float64
Counting-01                              float64
Counting-02                              float64
Place Value-01                           float64
Place Value-02                           float64
Addition and Subtraction-01              float64
Addition and Subtraction-02              float64
Multiplication and Division-01           float64
Multiplication and Division-02           float64
Year3_Writing_At_Risk                    float64
dtype: object
```

```
# Visualize correlations between features and the target variable 'Year3_Writing_At_Risk'
cor_relation=dataframe.corrwith(dataframe["Year3_Writing_At_Risk"])
cor_relation.plot(kind='bar')
matplt.show()
```



Model Training

```
# Split features and target for modeling
depent_axis = dataframe["Year3_Writing_At_Risk"]
independent_axis = dataframe.drop(["Year3_Writing_At_Risk"],axis=1)
```

Scaling Dataset from Standardize

```
# Standardize the features for better model performance
scaler = StandardScaler()
independent_axis = scaler.fit_transform(independent_axis)
independent_axis
```

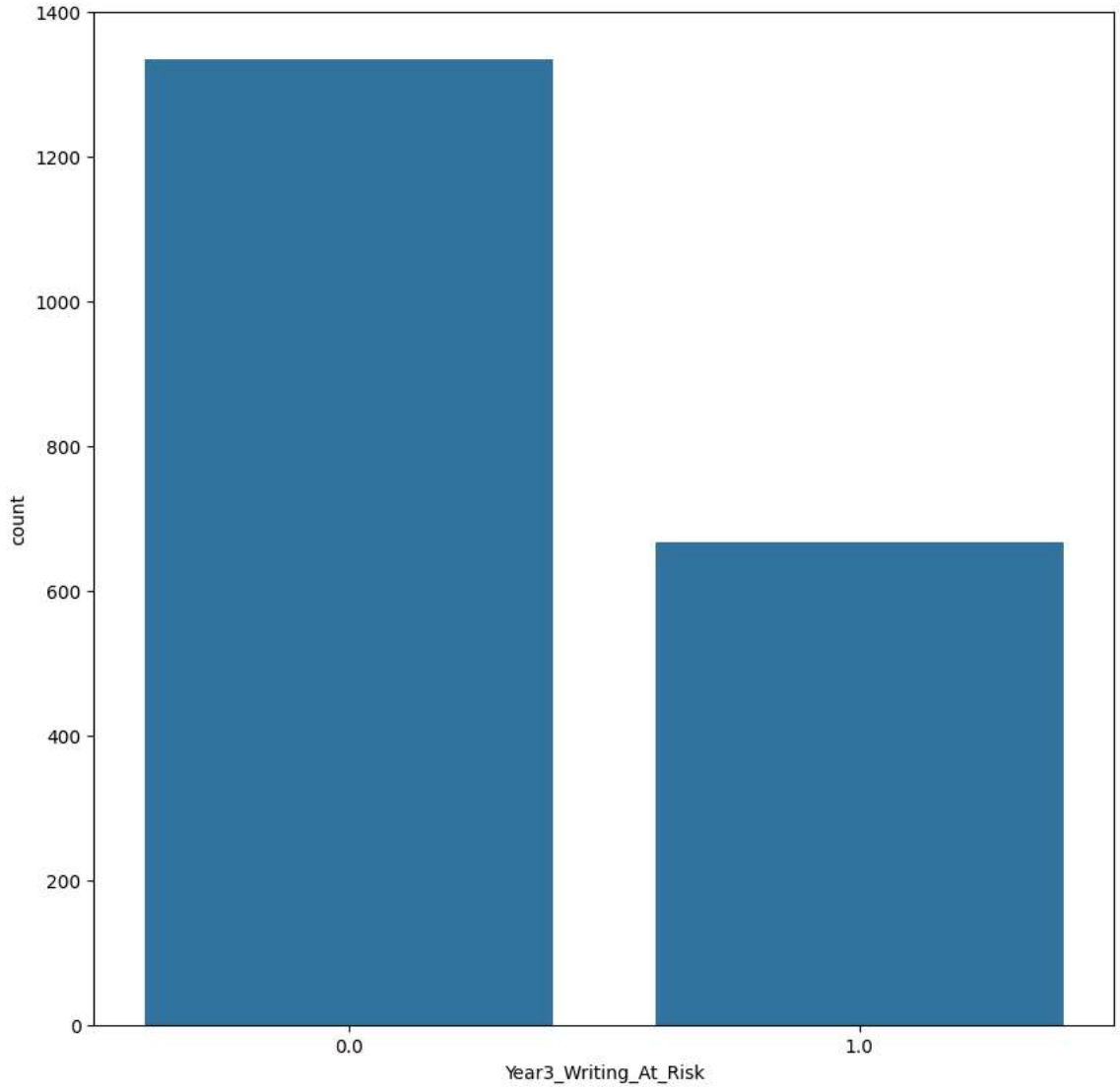
```
array([[-0.55347604,  0.66949677, -0.18617505, ...,  1.50799757,
         1.10357108,  0.26241365],
```

```
[-0.55347604,  0.66949677, -0.18617505, ...,  0.64257572,
 -1.28381903,  0.26241365],
[-0.55347604,  0.66949677, -0.18617505, ..., -1.08826797,
  1.10357108, -1.03345623],
...,
[-0.55347604,  0.66949677, -0.18617505, ..., -0.22284613,
 -1.28381903,  0.26241365],
[-0.55347604,  0.66949677, -0.18617505, ...,  1.50799757,
 -0.09012398,  0.26241365],
[ 1.80676294, -1.49365919, -0.18617505, ..., -0.22284613,
  1.10357108,  2.85415341]])
```

Split dataset into training and test sets

```
sns.countplot(x='Year3_Writing_At_Risk',data=dataframe)
```

```
<Axes: xlabel='Year3_Writing_At_Risk', ylabel='count'>
```

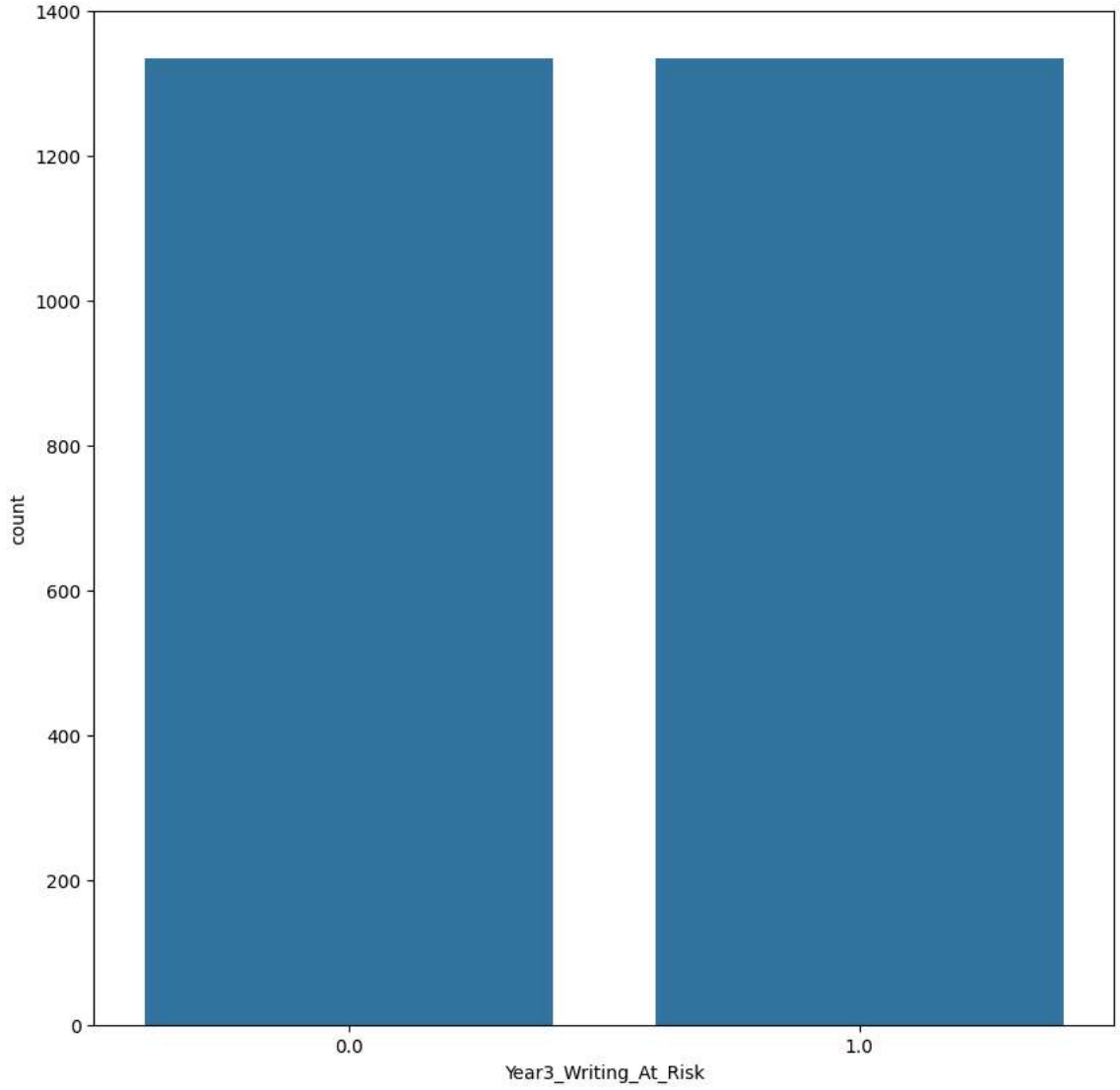



```
dataframe['Year3_Writing_At_Risk'].value_counts()

print("Presentage of people stroking = ")
len(dataframe[dataframe['Year3_Writing_At_Risk'] == 1])/len(dataframe)*100
```

```
Presentage of people stroking =
33.300000000000004
```

```
# Balance the target classes using SMOTE (Synthetic Minority Over-sampling)
independent_axis, depent_axis = SMOTE().fit_resample(independent_axis, depent_axis)

sns.countplot(x=depent_axis, data=dataframe)
```

```
<Axes: xlabel='Year3_Writing_At_Risk', ylabel='count'>
```



Split dataset into training and test sets

```
# Split data into training and testing sets
x_train,x_test,y_train,y_test = train_test_split(independent_axis,depent_axis,test_size=0.2,random_state=42)
```

RandomForest Classifier

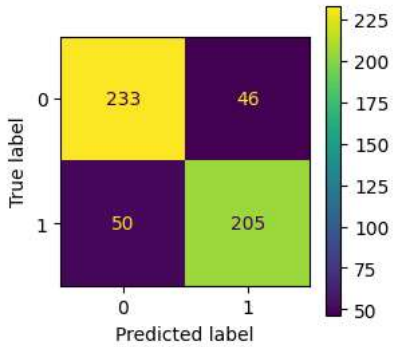
```
best_model = RandomForestClassifier(
    max_depth=30,
    max_features='sqrt',
    min_samples_leaf=2,
    min_samples_split=5,
    n_estimators=100,
    random_state=42,
    bootstrap=False
)
```

```
best_model.fit(x_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(bootstrap=False, max_depth=30, min_samples_leaf=2,
                       min_samples_split=5, random_state=42)
```

```
y_pred = best_model.predict(x_test)

r_accuracy = best_model.score(x_test, y_test)

cm = confusion_matrix(y_test, y_pred)

print('Presentage of Random Forest Classifier scores')
print(f'Random Forest Classifier Model accuracy\t: {r_accuracy}')
print(f'Presentage\t: {"{:.1%}".format(r_accuracy)}')
print(classification_report(y_test, y_pred))

print('Confusion Matrix:')
cmd = ConfusionMatrixDisplay(confusion_matrix=cm)

fig, ax = matplt.subplots(figsize=(3, 3))
cmd.plot(ax=ax)
matplt.show()
```

```
Presentage of Random Forest Classifier scores
Random Forest Classifier Model accuracy : 0.8202247191011236
Presentage      : 82.0%
              precision    recall  f1-score   support

         0.0       0.82      0.84      0.83       279
         1.0       0.82      0.80      0.81       255

    accuracy                           0.82       534
   macro avg       0.82      0.82      0.82       534
weighted avg       0.82      0.82      0.82       534

Confusion Matrix:
```



Save Random ForestClassifier best model

```
# Save the model as a pickle file
import pickle
model_filename = '/content/drive/My Drive/Colab Notebooks/Exam/random_forest_model.pkl'
with open(model_filename, 'wb') as file:
    pickle.dump(best_model, file)

print(f'Model saved as {model_filename}')
```

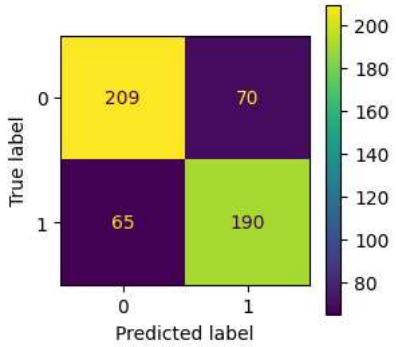
```
Model saved as /content/drive/My Drive/Colab Notebooks/Exam/random_forest_model.pkl
```

Support Vector Classifier

```
model = SVC()

model.fit(x_train, y_train)

y_pred = model.predict(x_test)

s_accuracy = model.score(x_test, y_test)

y_pred = model.predict(x_test)

cm = confusion_matrix(y_test, y_pred)
```

```
print('Presentage of Support vector Classifier scores')
print(f' Support vector Classifier Model accuracy\t: {s_accuracy}')
print(f'Presentage\t: {"{:.1%}".format(s_accuracy)}')
print(classification_report(y_test, y_pred))

print('Confusion Matrix - ')
cmd = ConfusionMatrixDisplay(confusion_matrix=cm);
figure, ax = matplt.subplots(figsize=(3,3))
cmd.plot(ax=ax)
```

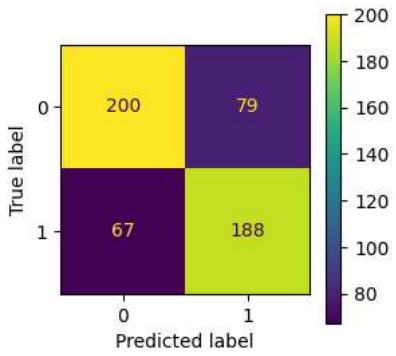
```
Presentage of Support vector Classifier scores
 Support vector Classifier Model accuracy       : 0.7471910112359551
Presentage      : 74.7%
              precision    recall  f1-score   support

         0.0       0.76      0.75      0.76       279
         1.0       0.73      0.75      0.74       255

    accuracy                           0.75       534
   macro avg       0.75      0.75      0.75       534
weighted avg       0.75      0.75      0.75       534

Confusion Matrix -
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x78629f1a3e80>
```



Decision Tree Classifier

```
model = DecisionTreeClassifier()

model.fit(x_train, y_train)

y_pred = model.predict(x_test)

d_accuracy = model.score(x_test, y_test)

cm = confusion_matrix(y_test, y_pred)

print('Decision Tree scores')
print(f'Decision Tree Model accuracy\t: {d_accuracy}')
print(f'Presentage\t: {"{:.1%}".format(d_accuracy)}')
print(classification_report(y_test, y_pred))

print('Confusion Matrix - ')
print(confusion_matrix(y_test, y_pred))

print('Confusion Matrix - ')
cmd = ConfusionMatrixDisplay(confusion_matrix=cm);
figure, ax = matplt.subplots(figsize=(3,3))
cmd.plot(ax=ax)
```

```
Decision Tree scores
Decision Tree Model accuracy    : 0.7265917602996255
Presentage      : 72.7%
              precision    recall  f1-score   support

         0.0       0.75      0.72      0.73       279
         1.0       0.70      0.74      0.72       255

    accuracy                           0.73       534
   macro avg       0.73      0.73      0.73       534
weighted avg       0.73      0.73      0.73       534

Confusion Matrix -
[[200  79]
 [ 67 188]]
Confusion Matrix -
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x78629f2ea620>
```



Logistic Regression

```
model = LogisticRegression()

model.fit(x_train, y_train)

y_pred = model.predict(x_test)

l_accuracy = model.score(x_test, y_test)

cm = confusion_matrix(y_test, y_pred)

print('Logistic Regression scores')
print(f'Logistic Regression Model accuracy\t: {l_accuracy}')
print(f'Percentage\t: {"{:.1%}".format(l_accuracy)}')
print(classification_report(y_test, y_pred))

print('Confusion Matrix - ')
print(confusion_matrix(y_test, y_pred))

print('Confusion Matrix - ')
cmd = ConfusionMatrixDisplay(confusion_matrix=cm);
figure, ax = matplt.subplots(figsize=(3,3))
cmd.plot(ax=ax)
```

```
Logistic Regression scores
Logistic Regression Model accuracy      : 0.6928838951310862
Percentage      : 69.3%
              precision    recall  f1-score   support

         0.0       0.71      0.71      0.71       279
         1.0       0.68      0.68      0.68       255

    accuracy                           0.69       534
   macro avg       0.69      0.69      0.69       534
weighted avg       0.69      0.69      0.69       534

Confusion Matrix -
[[197  82]
 [ 82 173]]
Confusion Matrix -
```

Visualize model accuracies

```
import pandas as pd
import matplotlib.pyplot as plt

model_accuracies = {
    'Random Forest': r_accuracy,
    'Support Vector Classifier': s_accuracy,
    'Decision Tree': d_accuracy,
    'Logistic Regression': l_accuracy
}

df = pd.DataFrame(list(model_accuracies.items()), columns=['Model', 'Accuracy'])

plt.figure(figsize=(10, 6))
plt.bar(df['Model'], df['Accuracy'], color=['skyblue', 'orange', 'green', 'red'])

plt.title('Model Accuracy Comparison', fontsize=16)
plt.xlabel('Model', fontsize=14)
plt.ylabel('Accuracy', fontsize=14)

for index, value in enumerate(df['Accuracy']):
    plt.text(index, value + 0.005, "{:.1%}".format(value), ha='center', fontsize=12)

plt.show()
```