

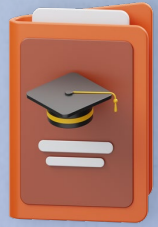
Advanced Programming  
23-2 Midterm Project

# Risk Factors of Cervical Cancer using Classification in Data Mining

IIE2110.01-00 | Team No.5

2019144060	Shin Seungwoo
2020147007	Lee Keunchan
2020147031	Lim Jonghyeok
2020147051	Jang Geon





Introduction

EDA

Preprocessing

PCA

Methodology

Experimental Results

Conclusion

# Risk Factors of Cervical Cancer using Classification in Data Mining

Nazim Razali et al  
2020 J. Phys.:  
Conf. Ser. 1529 022102

Nazim Razali, Salama A Mostafa,  
Aida Mustapha, Nurul Atieqah Ibrahim,  
Mohd Helmy Abd Wahab

Cervical cancer is the fourth most common cancer and often deadly and a big problem for women globally.

As technology advances, more research is being done on medical data.

This paper focuses on understanding risk factors for cervical cancer using data analysis techniques.





Introduction

EDA

Preprocessing

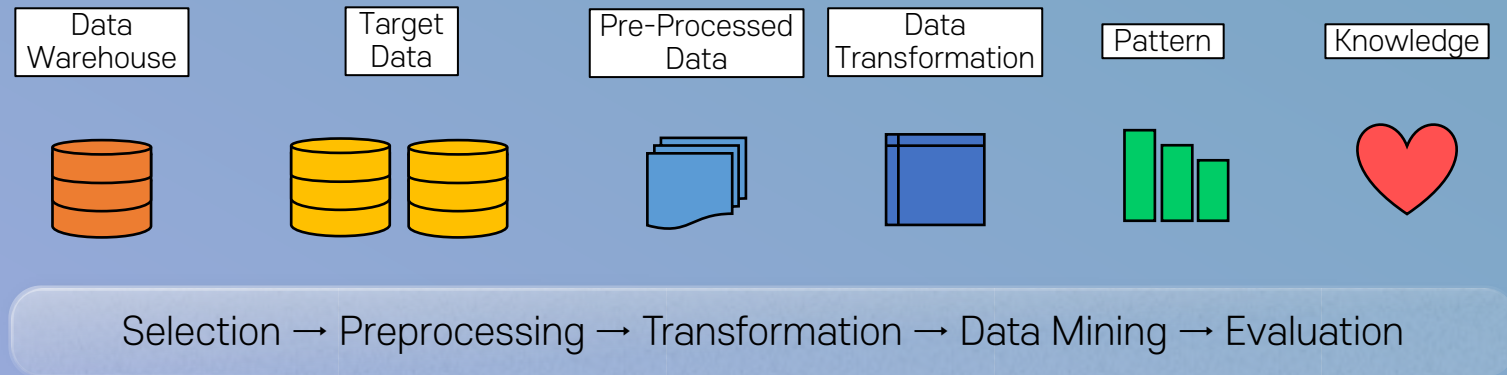
PCA

Methodology

Experimental Results

Conclusion

The experiment proposed in this paper followed the KDD(Knowledge Discovery in Data mining) process.



There are two important stages that need extra attention and concentration in this methodology which are

- 1) Data gathering and preparation stages (pre-processing)
- 2) Model building and evaluation stages.





Introduction

EDA

Preprocessing

PCA

Methodology

Experimental Results

Conclusion


The dataset used in this research is the cervical cancer risk factor extracted from UCI Machine Learning Repository.

**858** patients with  
**32** attributes and  
**1** target class



Our team also used the same dataset from UIC ML Repository\*.  
The goal of our project is to improve the accuracy presented in the paper.

\* <https://archive.ics.uci.edu/dataset/383/cervical+cancer+risk+factors>



Introduction
EDA
Preprocessing
PCA
Methodology
Experimental Results
Conclusion

## Import a csv file to explore the data:

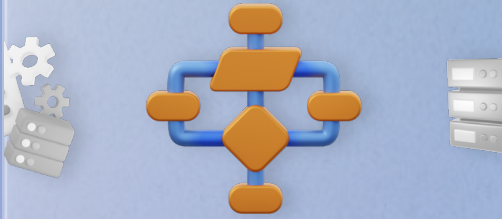
```
input_dir = r'C:\Users\risk_factors_cervical_cancer.csv'  
risk_factor_df = pd.read_csv(input_dir)  
risk_factor_df.head()
```

	Age	# of sexual partners	First sexual Intercourse	# of pregnancies	...	Hinselmann	Schiller	Citology	Biopsy
0	18	4.0	15.0	1.0	...	0	0	0	0
1	15	1.0	14.0	1.0	...	0	0	0	0
2	34	1.0	?	1.0	...	0	0	0	0
3	52	5.0	16.0	4.0	...	0	0	0	0
.	...	...	...	...	...	...	...	...	...

Total 858 rows x 36 Columns



We want to check the domain knowledge that indicators such as sexual partners, number of sexual relations, and contraception have a strong correlation with the y-value indicator.



Introduction

EDA

Preprocessing

PCA

Methodology

Experimental Results

Conclusion

## Check data type and missing values:

```
# Convert an object type column into a float type
```

```
for i in cols_to_convert:
    # Convert '?' into NaN
    risk_factor_df[cols_to_convert] =
    risk_factor_df[cols_to_convert].apply(pd.to_numeric, errors="coerce")
    # Enter missing values in each column as column-by-column median
    risk_factor_df[cols_to_convert] =
    risk_factor_df[cols_to_convert].fillna(risk_factor_df[cols_to_convert]
    .median())
```

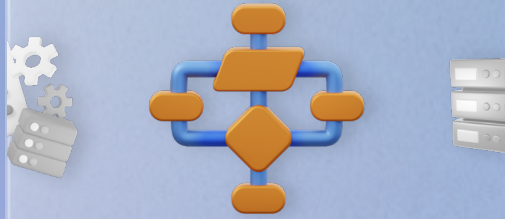
```
# Create a new input variable called age_cat and add age-specific
categories
```

```
# Make the total number of STDs as input variables by adding the values of
all STD input variables by row
```

```
# Make the sum of 4 cervical cancer test methods as a new input variable
```

As a method of data imputation for missing values, the researchers used the sample mean, whereas our project dealt with the median.





Introduction

EDA

Preprocessing

PCA

Methodology

Experimental Results

Conclusion

## Check data type and missing values:

```
risk_factor_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 858 entries, 0 to 857  
Data columns (total 36 columns):
```

#	Column	Dtype
0	Age	int64
1	Number of sexual partners	object
2	First sexual intercourse	object
3	Num of pregnancies	object
4	Smokes	object
(…result skip…)		
31	Dx	int64
32	Hinselmann	int64
33	Schiller	int64
34	Citology	int64
35	Biopsy	int64

dtypes: int64(10), object(26)

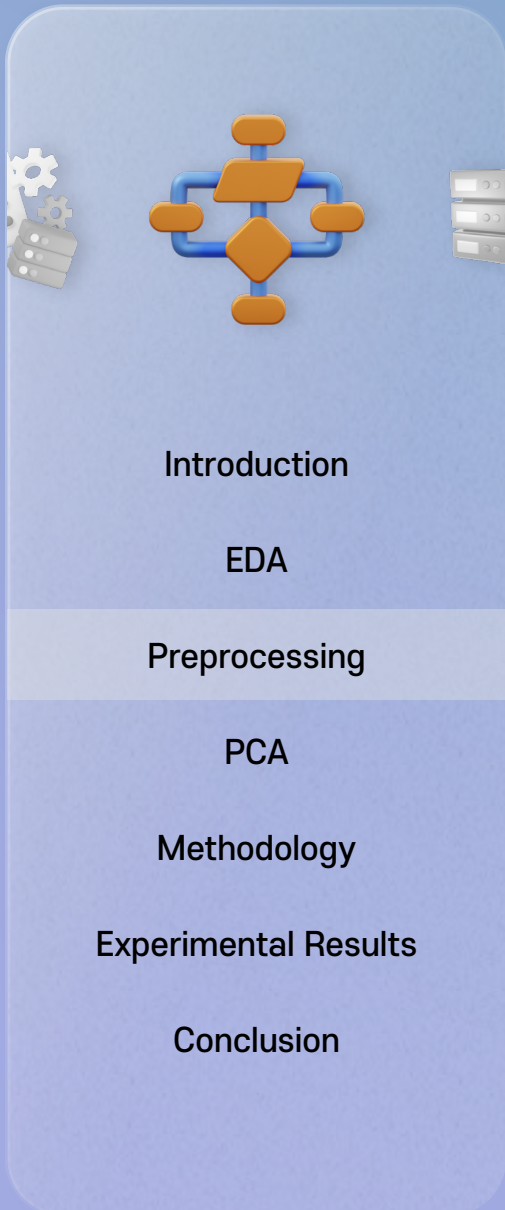
Before

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 858 entries, 0 to 857  
Data columns (total 39 columns):
```

#	Column	Dtype
0	Age	int64
1	Number of sexual partners	float64
2	First sexual intercourse	float64
3	Num of pregnancies	float64
4	Smokes	float64
(…result skip…)		
34	Citology	int64
35	Biopsy	int64
36	age_cat	object
37	total_std	float64
38	total_tests	int64

dtypes: float64(27), int64(11), object(1)

After



## Compute and visualize the correlation matrix:

```
n = 7
target = "Dx:Cancer"
corr = risk_factor_df.select_dtypes(include=[np.number]).corr()
top_corr_features = corr.nlargest(n + 1, target)[0:7].index
selected_corr_matrix = risk_factor_df[top_corr_features].corr()

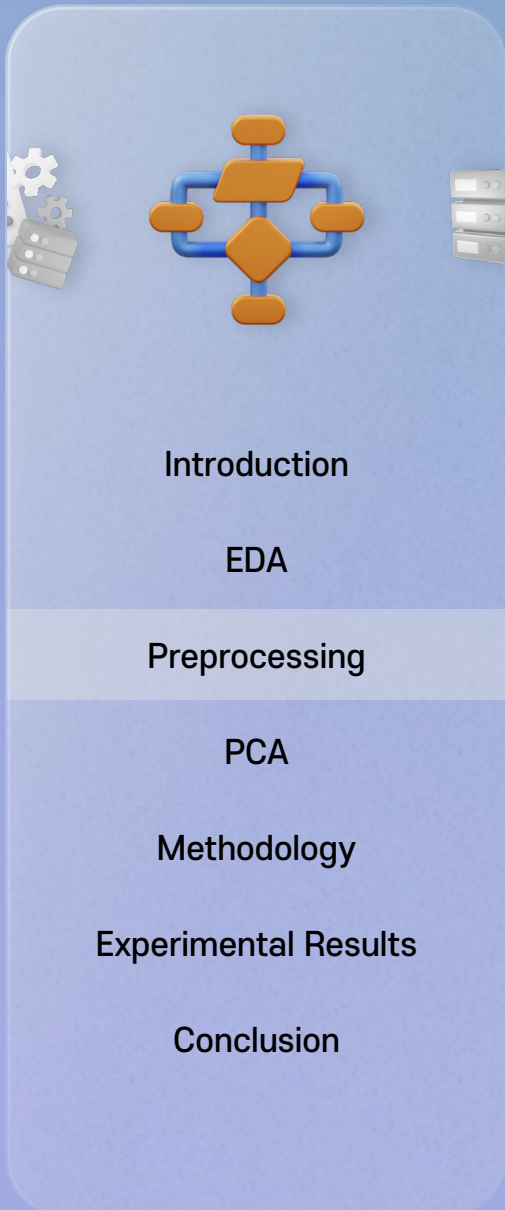
plt.figure(figsize=(10, 6))
sns.heatmap(selected_corr_matrix, annot=True, cmap='PuBu', fmt=".2f")
plt.title(f"Top {n} Features Correlated With {target.capitalize()}")
plt.show()
```

Dx:Cancer	1.00	0.89	0.67	0.33	0.18	0.16	0.16
Dx:HPV	0.89	1.00	0.62	0.33	0.18	0.16	0.16
Dx	0.67	0.62	1.00	0.14	0.14	0.16	0.10
STDs:HPV	0.33	0.33	0.14	1.00	-0.02	-0.01	-0.01
total_tests	0.18	0.18	0.14	-0.02	1.00	0.85	0.90
Biopsy	0.16	0.16	0.16	-0.01	0.85	1.00	0.73
Schiller	0.16	0.16	0.10	-0.01	0.90	0.73	1.00

The greater the number of sexual partners and the younger the age of sexual intercourse, the greater the risk of contracting HPV.

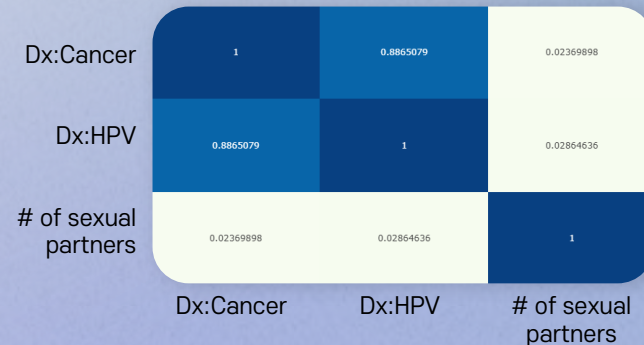
Weakened immunity and the development of HPV increase the risk of developing cervical cancer.



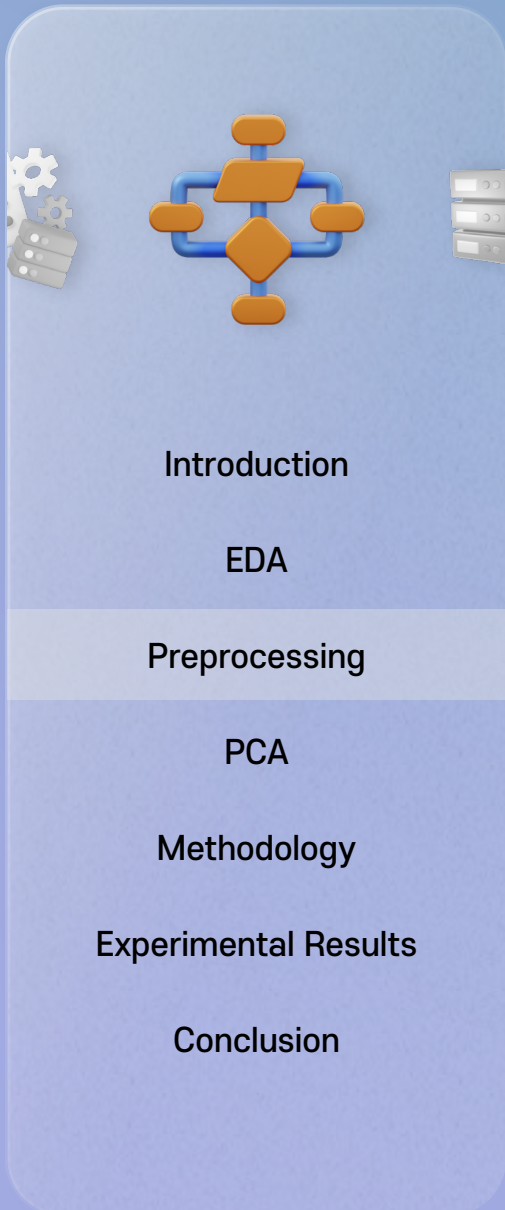


## Compute and visualize the correlation matrix:

```
diagnoses_num_partner_compare_cols = ['Dx:Cancer',  
                                       'Dx:HPV',  
                                       "Number of sexual partners",]  
corr_matrix = risk_factor_df[diagnoses_num_partner_compare_cols].corr()  
print(corr_matrix)  
diagnoses_num_partner_heatmap = px.imshow(corr_matrix,  
                                           aspect="auto",  
                                           color_continuous_scale="gnbu",  
                                           text_auto=True)  
diagnoses_num_partner_heatmap.show()
```



Number of sexual partners increases the risk of developing HPV  
cannot be verified from this data. (Correlation coefficient 0.02864..)

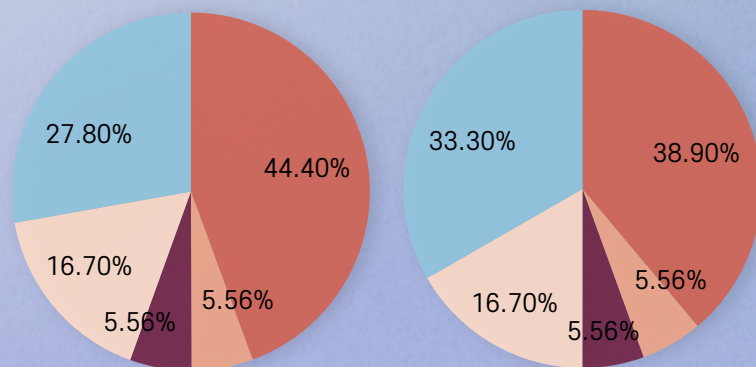


The cancer and HPV diagnosis rates for women by age are as follows:

```
fig = make_subplots(rows=1, cols=2, specs=[[{'type': 'domain'}], {'type': 'domain'}], subplot_titles=["Cancer", "HPV"])
fig.add_trace(go.Pie(labels=risk_factor_df["age_cat"], values=risk_factor_df["Dx:Cancer"], name="Cancer", marker_colors=px.colors.sequential.RdBu), 1, 1)
fig.add_trace(go.Pie(labels=risk_factor_df["age_cat"], values=risk_factor_df["Dx:HPV"], name="HPV", marker_colors=px.colors.sequential.RdBu), 1, 2)

# (...code skip...)
```

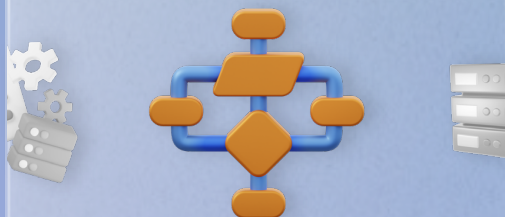
```
fig.show()
```



Proportion of women across age categories  
(Left : Cancer / Right : HPV)

● Teen ● 20's ● 30's ● 40's ● 50's ● 70+

Correlation for each disease is high regardless of age.



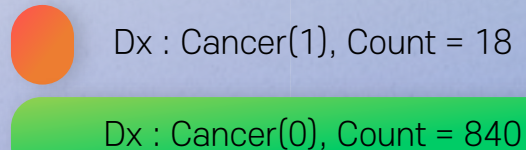
Introduction
EDA
Preprocessing
PCA
Methodology
Experimental Results
Conclusion

The results of resolving the imbalance by oversampling the data are as follows:

```
X = risk_factor_df.drop(["Dx:Cancer", "age_cat"], axis=1)
y = risk_factor_df["Dx:Cancer"].copy()
null_df = risk_factor_df.isnull().sum()
adasyn = ADASYN(random_state=42)
x_adasyn, y_adasyn = adasyn.fit_resample(X, y)
risk_factor_df = x_adasyn.join(y_adasyn)
risk_factor_df["age_cat"] = risk_factor_df["Age"].apply(age_cat)
dx_cancer = px.histogram(risk_factor_df, y="Dx:Cancer")

# (...code skip...)
```

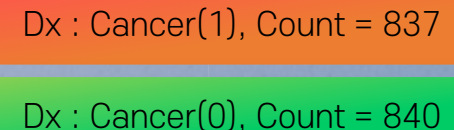
```
dx.cancer.show()
```



Dx : Cancer(1), Count = 18

Dx : Cancer(0), Count = 840

Before

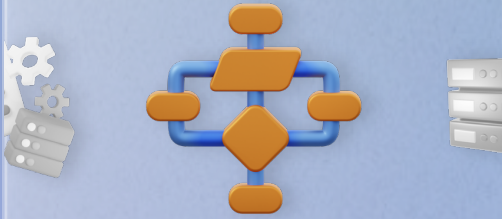


Dx : Cancer(1), Count = 837

Dx : Cancer(0), Count = 840

After





Introduction

EDA

Preprocessing

PCA

Methodology

Experimental Results

Conclusion

Divide the data into training and test sets as follows:

```
train_set = None
test_set = None
split = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=42)
for train_idx, test_idx in split.split(risk_factor_df,
risk_factor_df["age_cat"]):
    train_set = risk_factor_df.loc[train_idx]
    test_set = risk_factor_df.loc[test_idx]
cols_to_drop = ["age_cat", "total_std", "total_tests"]
for set_ in (train_set, test_set):
    for col in cols_to_drop:
        set_.drop(col, axis=1, inplace=True)
```

```
X_train = train_set.drop("Dx:Cancer", axis=1)
y_train = train_set["Dx:Cancer"].copy()
```

```
X_test = test_set.drop("Dx:Cancer", axis=1)
y_test = test_set["Dx:Cancer"].copy()
```

This is typically a required process for training and evaluating machine learning models.



Introduction

EDA

Preprocessing

PCA

Methodology

Experimental Results

Conclusion

Through PCA, 13 components with high explanatory power are extracted as follows:

```
XT = RobustScaler().fit_transform(X_train)
pca = PCA(n_components=0.99)
XT = pca.fit_transform(XT)
```

```
dimensions = px.bar(x=range(pca.n_components_),
                    y=pca.explained_variance_ratio_,
                    color_discrete_sequence=["darkslateblue"],
                    labels={"x": "PCA Feature", "y": "Explained Variance"})
exp_var_cumul = np.cumsum(pca.explained_variance_ratio_)

explained_variance = px.area(
    x=range(1, exp_var_cumul.shape[0] + 1),
    y=exp_var_cumul,
    labels={"x": "# Components", "y": "Explained Variance"},
    color_discrete_sequence=["dodgerblue"]
)
```

- Calculate explanatory variable proportions
- Output the top features of principal components of PCA
- Normalize using RobustScaler and Reduce the dimensionality



Introduction

EDA

Preprocessing

PCA

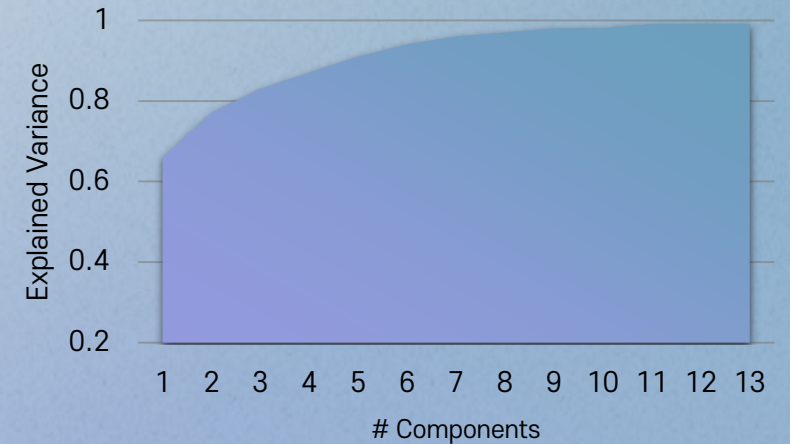
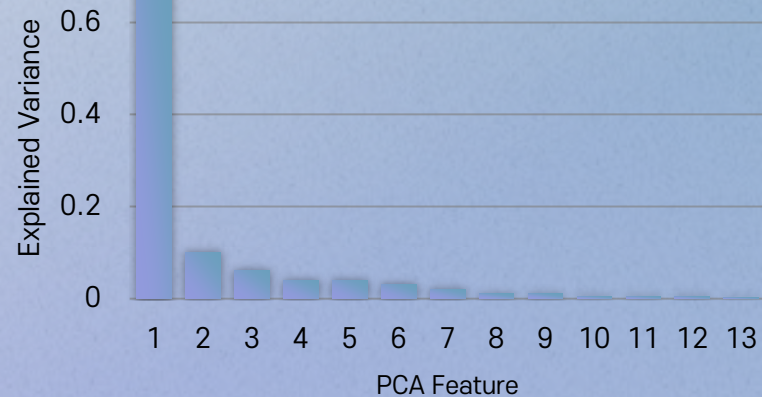
Methodology

Experimental Results

Conclusion

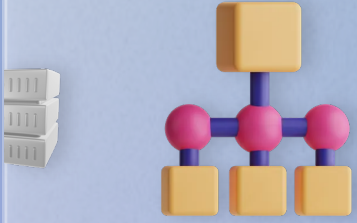
Set the PCA-processed x\_train data as input data for each model as follows:

```
Dimensions.show() # Visualization of explanatory variable ratio  
Explained_variance.show() # ~ for each principal component
```



```
pipeline = Pipeline([  
    ("scaler", RobustScaler()),  
    ("pca", PCA(n_components=13))  
)  
X_train = pipeline.fit_transform(X_train)  
X_test = pipeline.transform(X_test)
```





Introduction

EDA

Preprocessing

PCA

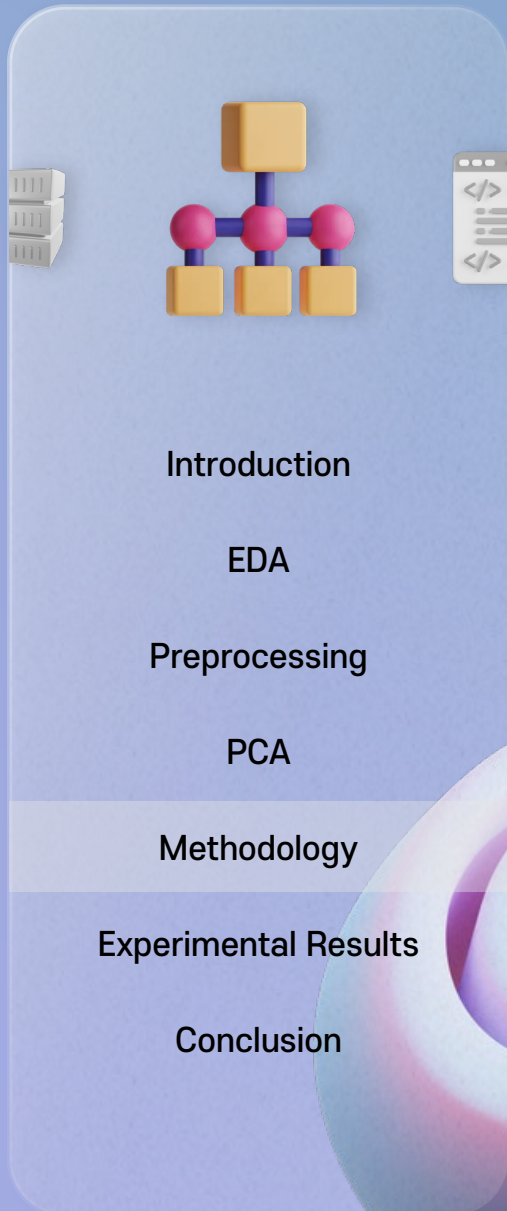
Methodology

Experimental Results

Conclusion

## (1) Logistic Regression

- Used for binary classification, not regression.
- Models the probability of a point belonging to a class using a logistic function.
- Simple and interpretable, making it a good starting point for classification problems.

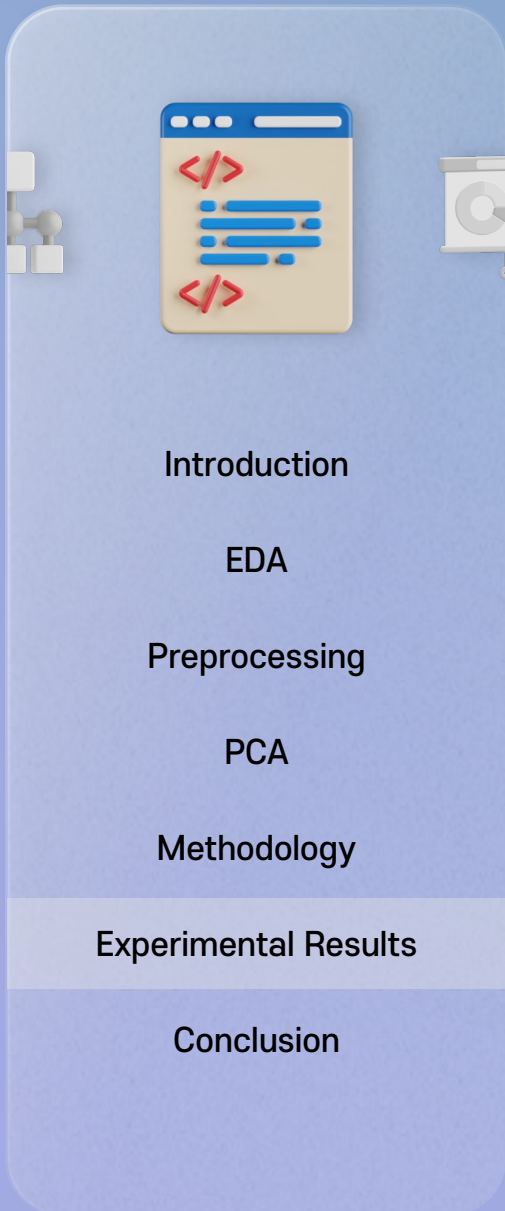


## (2) K-Nearest Neighbors Classifier

- Instance-based learning algorithm.
- Classifies new points based on majority class of "k" nearest neighbors.
- Easy to understand and implement, but can be computationally expensive with large datasets.

## (3) Support Vector Classifier

- Powerful algorithm for classification and regression problems.
- Aims to find the optimal hyperplane for class separation by maximizing the margin.
- Effective in high-dimensional spaces and can handle non-linear boundaries with kernel functions.



Modeling is performed with the data as follows:



```
# (...code skip...)

estimators = [
    ("LogisticRegression", logreg_cv),
    ("KNeighborsClassifier", knn_clf_cv),
    ("SupportVectorClassifier", svm_clf_cv)]

for i in range(0, len(estimators)):
    clf_name = estimators[i][0]
    clf = estimators[i][1]
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    est_name.append(estimators[i][0])
    est_acc.append(accuracy_score(y_test, y_pred))
    scores = precision_recall_fscore_support
              (y_test, y_pred, average="weighted")
    precision_score.append(scores[0])
    recall_score.append(scores[1])
    f1score.append(scores[2])
    est_conf_matrix.append(confusion_matrix(y_test, y_pred))

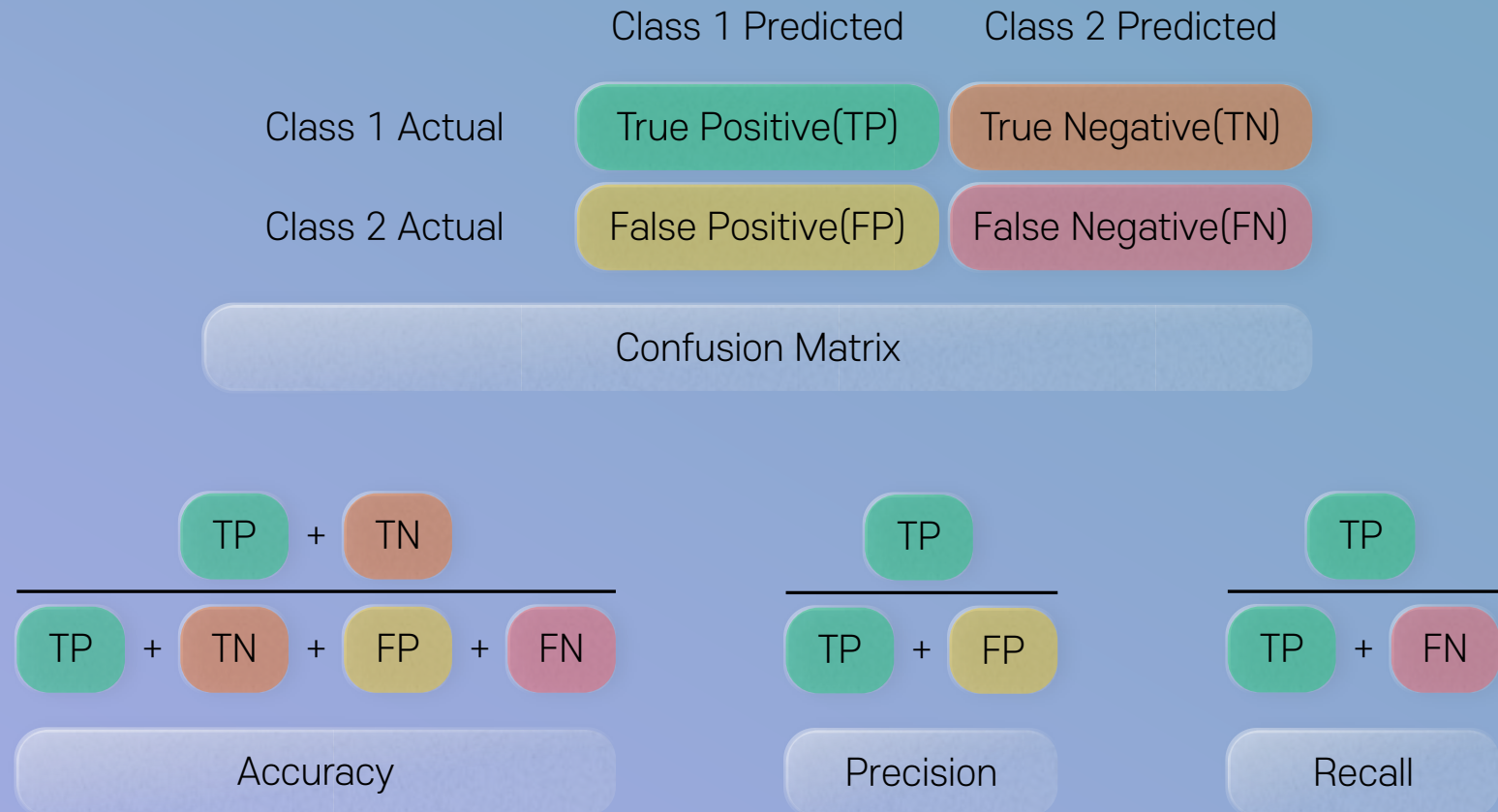
# (...code skip...)
```

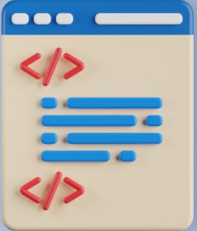




Introduction
EDA
Preprocessing
PCA
Methodology
Experimental Results
Conclusion

Evaluate performance, including model accuracy, using the following evaluation metrics as risk levels:





Introduction
EDA
Preprocessing
PCA
Methodology
Experimental Results
Conclusion

The performance of each algorithm in our team's project is as follows:

```
color_scales = ["agsunset", "teal", "purp", "viridis"]
for i in range(0, len(est_conf_matrix)):
    heatmap = px.imshow(est_conf_matrix[i], aspect="auto",
                        text_auto=True,
                        color_continuous_scale=color_scales[i])
    heatmap.update_layout(title = est_name[i])
    heatmap.update_xaxes(title="Predicted")
    heatmap.update_yaxes(title="Actual")
    heatmap.show()
```

TP  
173

TN  
2

FP  
0

FN  
161

Logistic  
Regression

TP  
163

TN  
12

FP  
1

FN  
160

K-nearest  
Neighbors  
Classifier

TP  
173

TN  
2

FP  
0

FN  
161

Support  
Vector Classifier

The performance of each classifier in our team's project improved compared to the paper as follows:

`Acc_comparison.show()`

Introduction

EDA

Preprocessing

PCA

Methodology

Experimental Results

Conclusion

76.9 ▶ **99.4** (29.3%▲)

77.1 ▶ **99.4** (28.9%▲)

76.9 ▶ **99.4** (29.3%▲)

Logistic Regression

92.6 ▶ **96.1** (3.78%▲)

92.6 ▶ **96.3** (4.00%▲)

92.6 ▶ **96.1** (3.78%▲)

K-nearest Neighbors Classifier

74.7 ▶ **99.4** (33.1%▲)

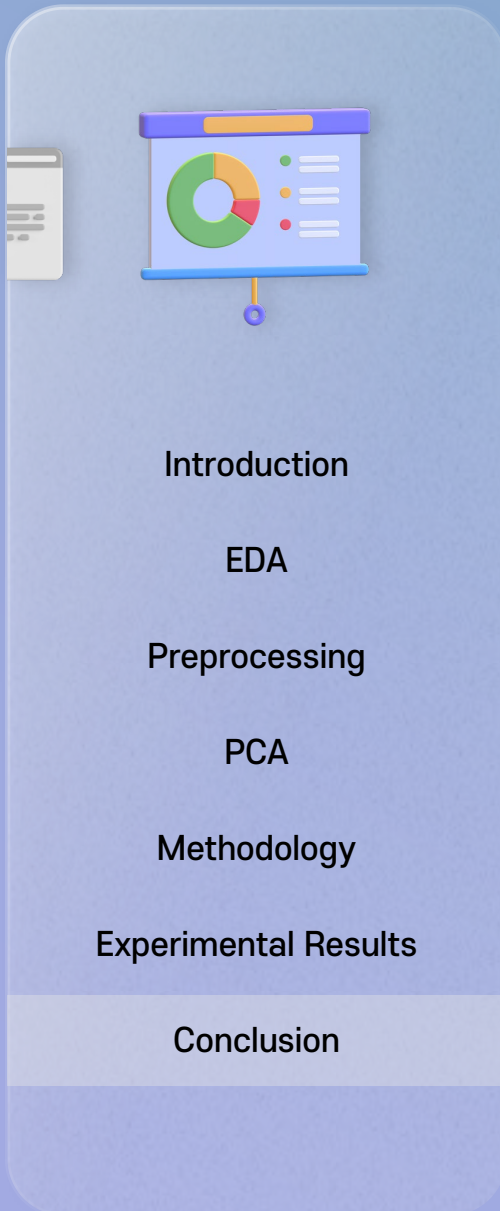
74.7 ▶ **99.4** (33.1%▲)

74.6 ▶ **99.4** (33.2%▲)

Support Vector Classifier

● Accuracy Score(%) ● Prediction Score(%) ● Recall Score(%)

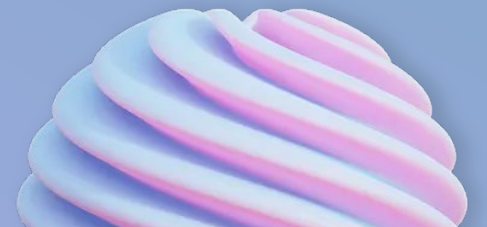




The differences from the modeling method performed in the paper are as follows:

- A difference was made in the way missing values were handled (median instead of sample mean).
- While the paper used the SMOTE to generate the same number of synthetic samples from all minority classes, our team applied ADASYN to generate more samples to ensure that the minority class was not ignored.
- By applying PCA, we attempted to explain the main variability of the data, and can expect to remove noise and improve independence of the data.

These factors changed the performance of the model compared to previous studies, and most of them resulted in significant improvements.



```
print("Thank you!")
```

```
import turtle

screen = turtle.Screen()
screen.bgcolor("white")

t = turtle.Turtle()
t.pensize(3)
t.color("red")
t.begin_fill()

t.left(50)
t.forward(133)
t.circle(50, 200)
t.right(140)
t.circle(50, 200)
t.forward(133)

t.end_fill()
turtle.done()
```

Thank you!

