CENG542
KNOWLEDGE DISCOVERY
TERM PROJECT PRESENTATION

PREDICTING ALCOHOL CONSUMPTION AMONG STUDENT

Group Members

Burak Topçu 283078027

Güliz Akkaya 283078008

İsmail Cebeci 272001030

OVERVIEW

Introduction

Methodology

Experiments and Results

Conclusion

References

Introduction

- The uncontrolled alcohol consumption among students is a significant issue.
- As a result of high level of alcohol consumption, students may have some problems in daily and academic life. For this reason, it is important to analyze the factors that are related to alcohol consumption among students and this information may be utilized to find solutions for this issue.

Introduction

- In this project, In this project, we investigate the features of students that are related to alcohol consumption using Pearson correlation method and feature selection at first.
- Afterwards, 4 different outlier detection methods such as Isolation Forest, Elliptic Envelope, Local Outlier Factor and OneClass SVM are utilized to find outliers in the data.
- We perform classification on student alcohol consumption dataset [1] using different algorithms known as Support Vector Machines (SVM), Stochastic Gradient Descent (SGD), Random Forest (RF), Gradient Boosting (GB), Decision Tree (DT) and AdaBoost.

Methodology

- Data preprocessing
- Feature selection
- Outlier detection
- Classification

Methodology - Preprocessing

- The dataset collected by Paulo Cortez and Alice Silva [1] for student performance predictions is utilized.
- The values of some attributes are not numeric in the dataset.
- These attribute values are converted to numerical values to process data appropriately.

i. 1 for 'yes'

ii. 0 for 'no'

Methodology – Feature Selection

- Pearson correlation method is used to observe the relationships between the features.
- Some attributes such as family size and education level of parents are removed from dataset according to correlation values.
- Feature selection is performed by assigning Random Forest Classifier as estimator.

Methodology – Outlier Detection

- 4 different methods are used to detect outliers

i. Isolation Forest

ii. Elliptic Envelope

iii. Local Outlier Factor

iv. One Class SVM

- Most successful outlier detector is the Elliptic Envelope method
- 65 outlier samples are removed to enhance classification accuracy results

Methodology - Classification

- 6 different algorithms are used for classification

  i. Support Vector Machines

  ii. Stochastic Gradient Descent

  iii. Random Forest

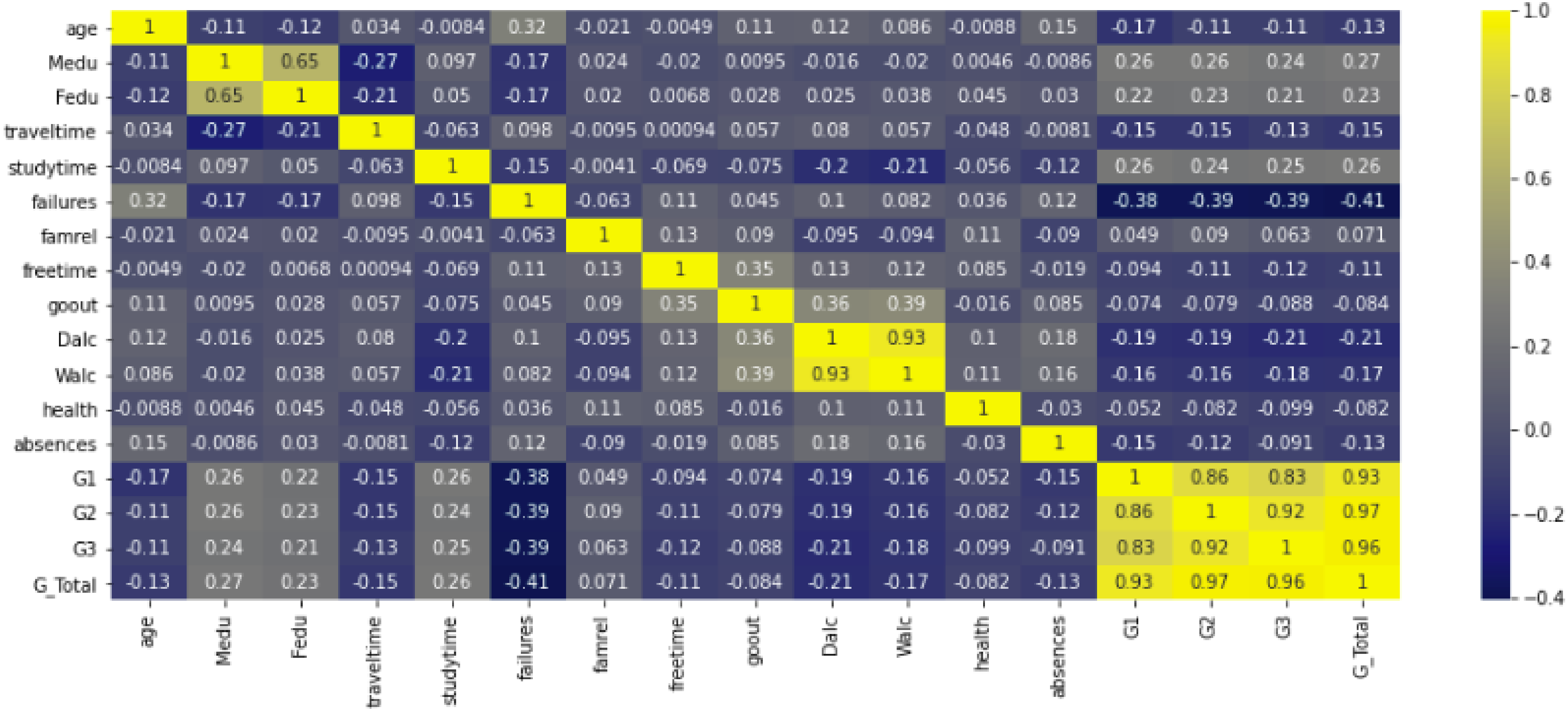
  iv. Gradient Boosting

  v. Decision Tree

  vi. AdaBoost

- Support Vector Machines algorithm produces highest accuracy results

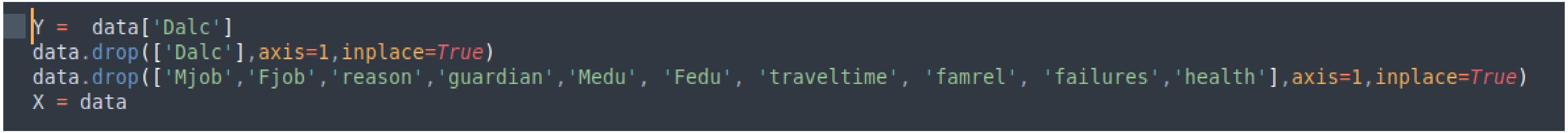
Experiments and Results - Feature selection



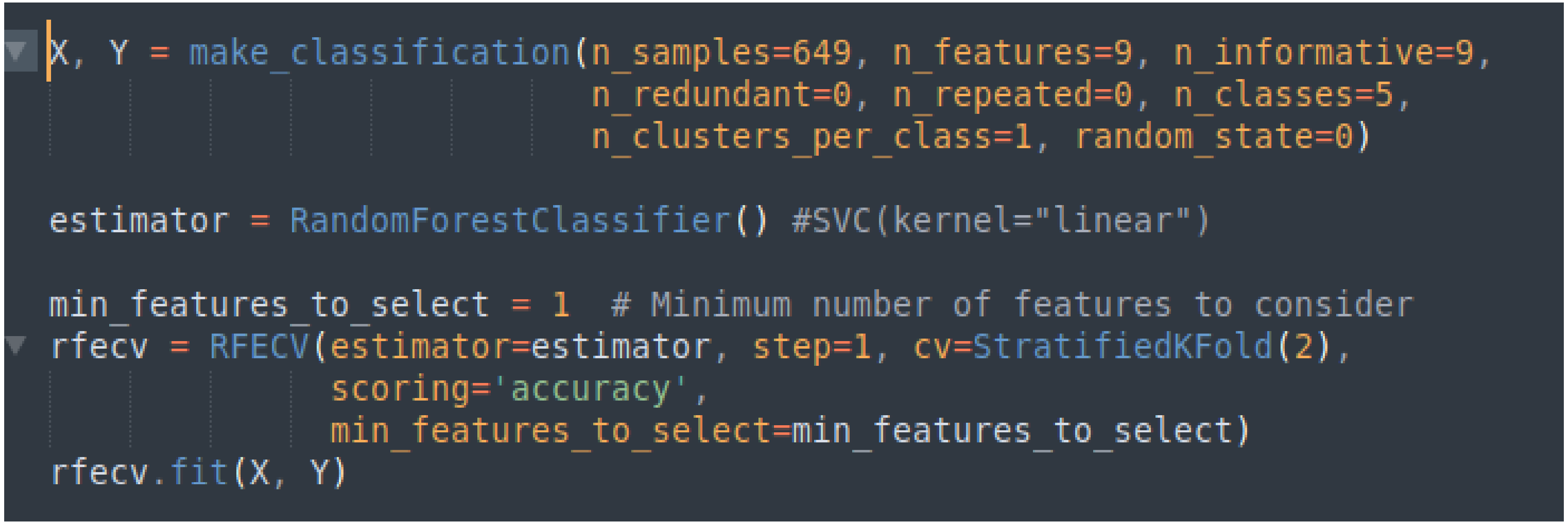
	age	Medu	Fedu	traveltime	studytime	failures	famrel	freetime	goout	Dalc	Walc	health	absences	G1	G2	G3	G_Total
age	1	-0.11	-0.12	0.034	-0.0084	0.32	-0.021	-0.0049	0.11	0.12	0.086	-0.0088	0.15	-0.17	-0.11	-0.11	-0.13
Medu	-0.11	1	0.65	-0.27	0.097	-0.17	0.024	-0.02	0.0095	-0.016	-0.02	0.0046	-0.0086	0.26	0.26	0.24	0.27
Fedu	-0.12	0.65	1	-0.21	0.05	-0.17	0.02	0.0068	0.028	0.025	0.038	0.045	0.03	0.22	0.23	0.21	0.23
traveltime	0.034	-0.27	-0.21	1	-0.063	0.098	-0.0095	0.00094	0.057	0.08	0.057	-0.048	-0.0081	-0.15	-0.15	-0.13	-0.15
studytime	-0.0084	0.097	0.05	-0.063	1	-0.15	-0.0041	-0.069	-0.075	-0.2	-0.21	-0.056	-0.12	0.26	0.24	0.25	0.26
failures	0.32	-0.17	-0.17	0.098	-0.15	1	-0.063	0.11	0.045	0.1	0.082	0.036	0.12	-0.38	-0.39	-0.39	-0.41
famrel	-0.021	0.024	0.02	-0.0095	-0.0041	-0.063	1	0.13	0.09	-0.095	-0.094	0.11	-0.09	0.049	0.09	0.063	0.071
freetime	-0.0049	-0.02	0.0068	0.00094	-0.069	0.11	0.13	1	0.35	0.13	0.12	0.085	-0.019	-0.094	-0.11	-0.12	-0.11
goout	0.11	0.0095	0.028	0.057	-0.075	0.045	0.09	0.35	1	0.36	0.39	-0.016	0.085	-0.074	-0.079	-0.088	-0.084
Dalc	0.12	-0.016	0.025	0.08	-0.2	0.1	-0.095	0.13	0.36	1	0.93	0.1	0.18	-0.19	-0.19	-0.21	-0.21
Walc	0.086	-0.02	0.038	0.057	-0.21	0.082	-0.094	0.12	0.39	0.93	1	0.11	0.16	-0.16	-0.16	-0.18	-0.17
health	-0.0088	0.0046	0.045	-0.048	-0.056	0.036	0.11	0.085	-0.016	0.1	0.11	1	-0.03	-0.052	-0.082	-0.099	-0.082
absences	0.15	-0.0086	0.03	-0.0081	-0.12	0.12	-0.09	-0.019	0.085	0.18	0.16	-0.03	1	-0.15	-0.12	-0.091	-0.13
G1	-0.17	0.26	0.22	-0.15	0.26	-0.38	0.049	-0.094	-0.074	-0.19	-0.16	-0.052	-0.15	1	0.86	0.83	0.93
G2	-0.11	0.26	0.23	-0.15	0.24	-0.39	0.09	-0.11	-0.079	-0.19	-0.16	-0.082	-0.12	0.86	1	0.92	0.97
G3	-0.11	0.24	0.21	-0.13	0.25	-0.39	0.063	-0.12	-0.088	-0.21	-0.18	-0.099	-0.091	0.83	0.92	1	0.96
G_Total	-0.13	0.27	0.23	-0.15	0.26	-0.41	0.071	-0.11	-0.084	-0.21	-0.17	-0.082	-0.13	0.93	0.97	0.96	1

Pearson Correlations between the features.

Experiments and Results - Feature selection



```
Y =  data['Dalc']
data.drop(['Dalc'],axis=1,inplace=True)
data.drop(['Mjob','Fjob','reason','guardian','Medu', 'Fedu', 'traveltime', 'famrel', 'failures','health'],axis=1,inplace=True)
X = data
```

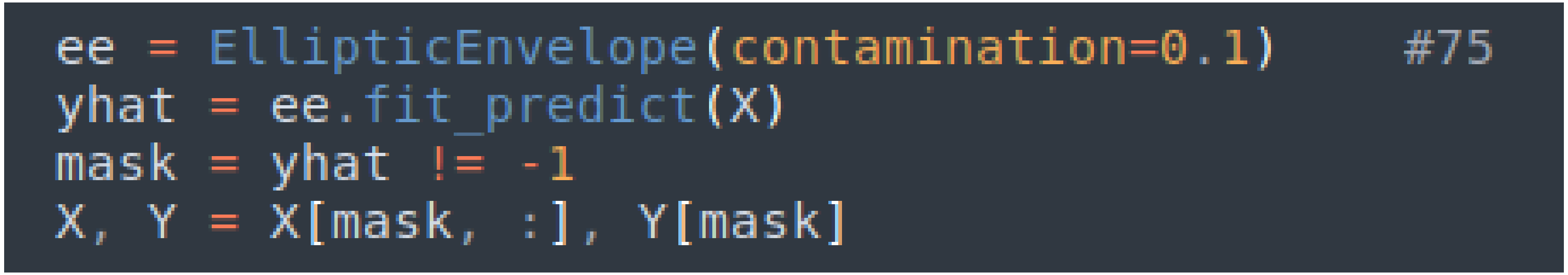


```
X, Y = make_classification(n_samples=649, n_features=9, n_informative=9,
                           n_redundant=0, n_repeated=0, n_classes=5,
                           n_clusters_per_class=1, random_state=0)

estimator = RandomForestClassifier() #SVC(kernel="linear")

min_features_to_select = 1  # Minimum number of features to consider
rfecv = RFECV(estimator=estimator, step=1, cv=StratifiedKFold(2),
              scoring='accuracy',
              min_features_to_select=min_features_to_select)
rfecv.fit(X, Y)
```

- Age, study time, go out level, number of school absences, course grades and weekend alcohol consumption level are most related attributes.

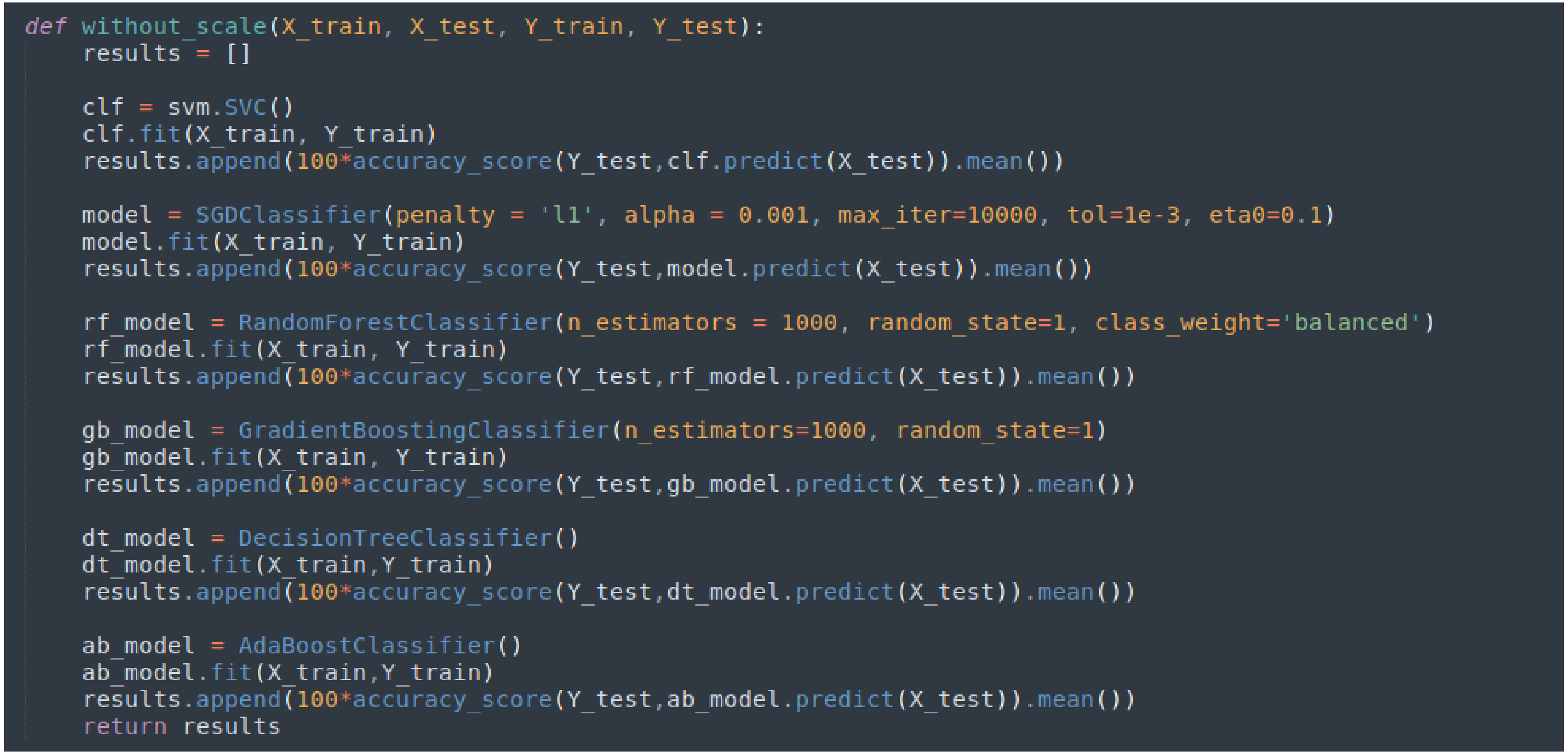
Experiments and Results - Outlier Detection

- Some of the samples that have irrelevant features are detected and dropped from the data set to prevent disturbing classification methods.
- With EllipticEnvelope algorithm with contamination = 0.1



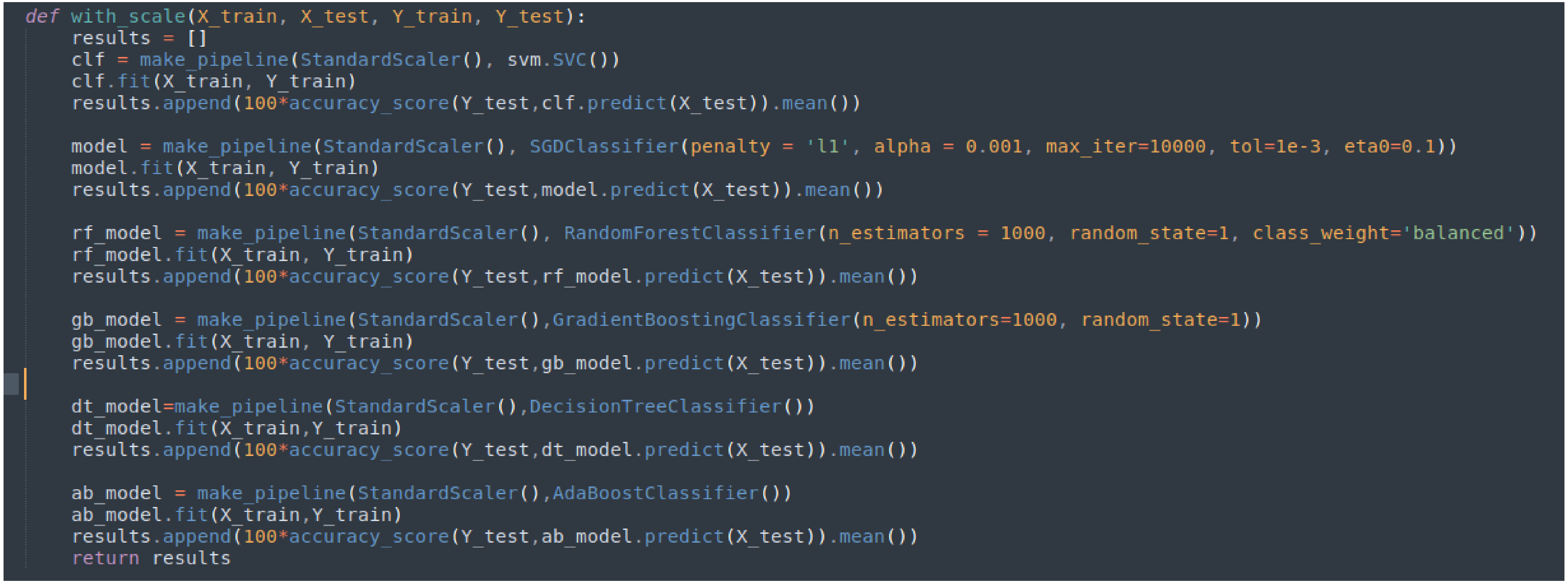
```
ee = EllipticEnvelope(contamination=0.1)    #75
yhat = ee.fit_predict(X)
mask = yhat != -1
X, Y = X[mask, :], Y[mask]
```

Experiments and Results



```
def without_scale(X_train, X_test, Y_train, Y_test):
    results = []

    clf = svm.SVC()
    clf.fit(X_train, Y_train)
    results.append(100*accuracy_score(Y_test,clf.predict(X_test)).mean())

    model = SGDClassifier(penalty = 'l1', alpha = 0.001, max_iter=10000, tol=1e-3, eta0=0.1)
    model.fit(X_train, Y_train)
    results.append(100*accuracy_score(Y_test,model.predict(X_test)).mean())

    rf_model = RandomForestClassifier(n_estimators = 1000, random_state=1, class_weight='balanced')
    rf_model.fit(X_train, Y_train)
    results.append(100*accuracy_score(Y_test,rf_model.predict(X_test)).mean())

    gb_model = GradientBoostingClassifier(n_estimators=1000, random_state=1)
    gb_model.fit(X_train, Y_train)
    results.append(100*accuracy_score(Y_test,gb_model.predict(X_test)).mean())

    dt_model = DecisionTreeClassifier()
    dt_model.fit(X_train,Y_train)
    results.append(100*accuracy_score(Y_test,dt_model.predict(X_test)).mean())

    ab_model = AdaBoostClassifier()
    ab_model.fit(X_train,Y_train)
    results.append(100*accuracy_score(Y_test,ab_model.predict(X_test)).mean())
    return results
```

Experiments and Results



```
def with_scale(X_train, X_test, Y_train, Y_test):
    results = []
    clf = make_pipeline(StandardScaler(), svm.SVC())
    clf.fit(X_train, Y_train)
    results.append(100*accuracy_score(Y_test,clf.predict(X_test)).mean())

    model = make_pipeline(StandardScaler(), SGDClassifier(penalty = 'l1', alpha = 0.001, max_iter=10000, tol=1e-3, eta0=0.1))
    model.fit(X_train, Y_train)
    results.append(100*accuracy_score(Y_test,model.predict(X_test)).mean())

    rf_model = make_pipeline(StandardScaler(), RandomForestClassifier(n_estimators = 1000, random_state=1, class_weight='balanced'))
    rf_model.fit(X_train, Y_train)
    results.append(100*accuracy_score(Y_test,rf_model.predict(X_test)).mean())

    gb_model = make_pipeline(StandardScaler(),GradientBoostingClassifier(n_estimators=1000, random_state=1))
    gb_model.fit(X_train, Y_train)
    results.append(100*accuracy_score(Y_test,gb_model.predict(X_test)).mean())

    dt_model=make_pipeline(StandardScaler(),DecisionTreeClassifier())
    dt_model.fit(X_train,Y_train)
    results.append(100*accuracy_score(Y_test,dt_model.predict(X_test)).mean())

    ab_model = make_pipeline(StandardScaler(),AdaBoostClassifier())
    ab_model.fit(X_train,Y_train)
    results.append(100*accuracy_score(Y_test,ab_model.predict(X_test)).mean())
    return results
```

Experiments and Results

- We have made 3 experiments:

1) Including the neither feature selection nor the outlier detection

	Without Standard Scale	With Standard Scale
SVM	0.70	0.69
SGD	0.68	0.68
RF	0.70	0.70
GB	0.65	0.66
DT	0.56	0.57
AB	0.48	0.48

Experiments and Results

2) Including just the feature selection (without the outlier detection)

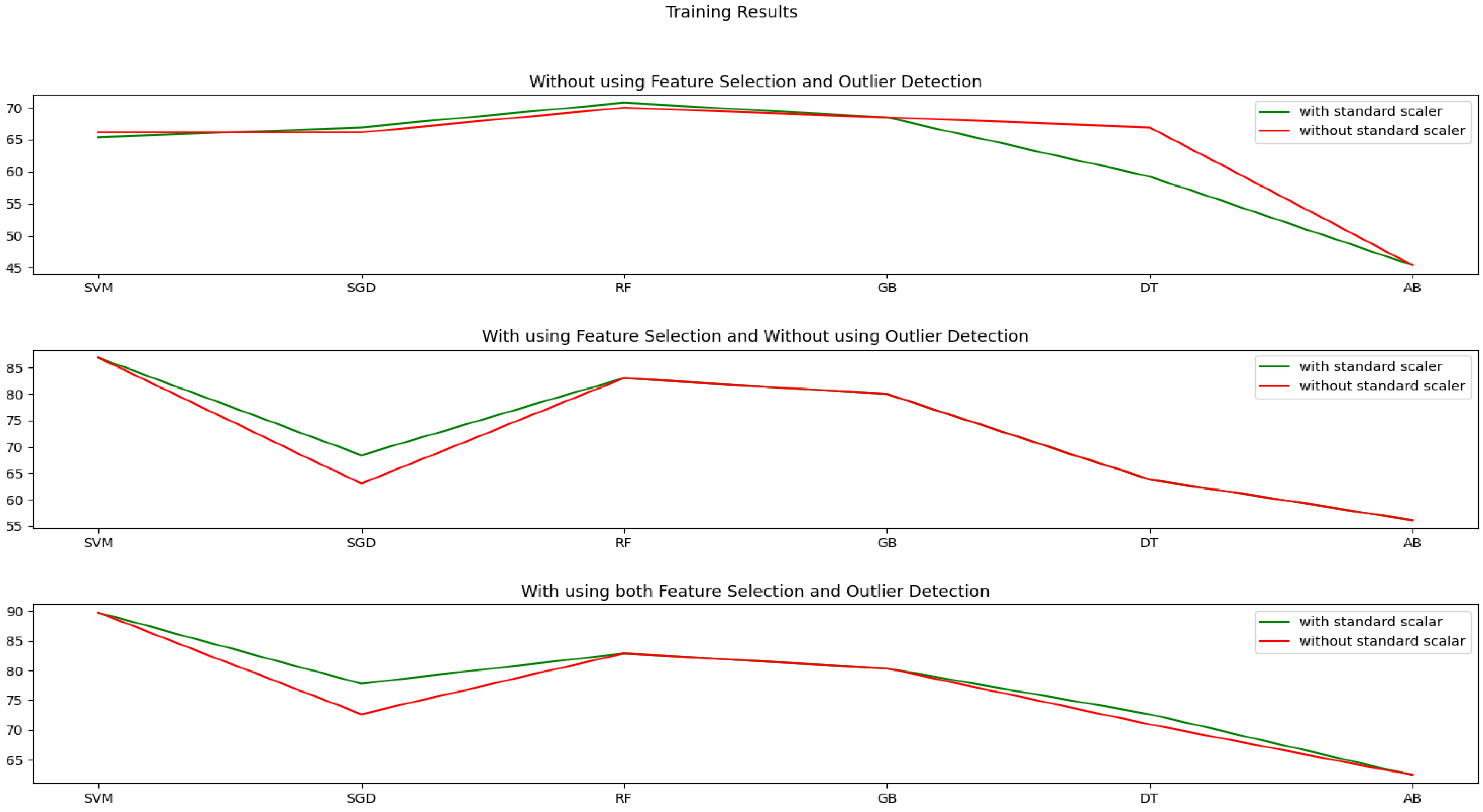
	Without Standard Scale	With Standard Scale
SVM	0.86	0.86
SGD	0.60	0.68
RF	0.80	0.80
GB	0.70	0.70
DT	0.59	0.6
AB	0.53	0.53

Click to add text

Experiments and Results

3) Including both the feature selection and the outlier detection

	Without Standard Scale	With Standard Scale
SVM	0.93	0.93
SGD	0.73	0.76
RF	0.85	0.85
GB	0.78	0.78
DT	0.52	0.58
AB	0.62	0.62



Training Results

Without using Feature Selection and Outlier Detection

With using Feature Selection and Without using Outlier Detection

With using both Feature Selection and Outlier Detection

Conclusion

- We investigated the relations between alcohol consumption among students and their traits such as age, family education, educational success and so on.
- The experiments where feature selection and outlier detection methods are applied consists of 6 different algorithms and 8 selected features.
- As a result, Support Vector Machines algorithm results in higher accuracy compared to other algorithms. The final classification accuracy is 93% with the SVM.

References

[1] Cortez, Paulo & Silva, Alice. (2008). Using data mining to predict secondary school student performance. EUROSIS.

[2] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830.