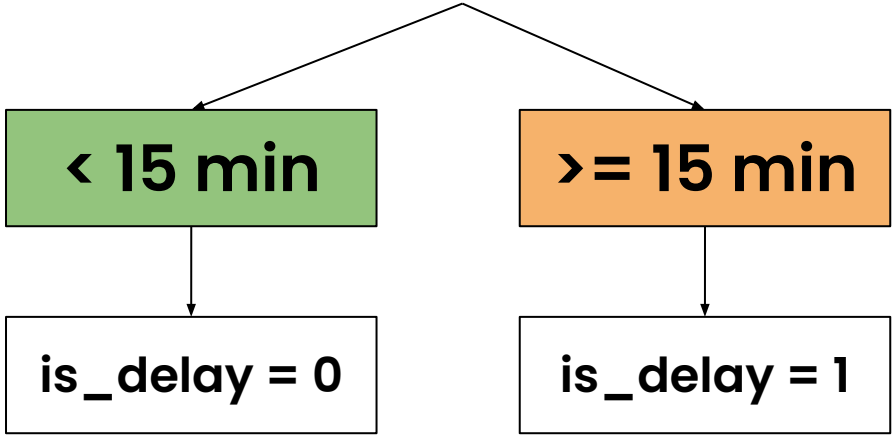
Flight delay prediction in JFK airport using machine learning classifiers

Carlos Moreno Tavira

May 2022

Objectiu del projecte

- Predir el *delay* dels vols que surten de New York



Federal Aviation Administration

	dep_delay	is_delay
0	-1	0
1	-7	0
2	40	1
3	-2	0
4	-4	0
...	...	...
28813	2	0
28814	2	0
28815	283	1
28816	5	0
28817	-1	0

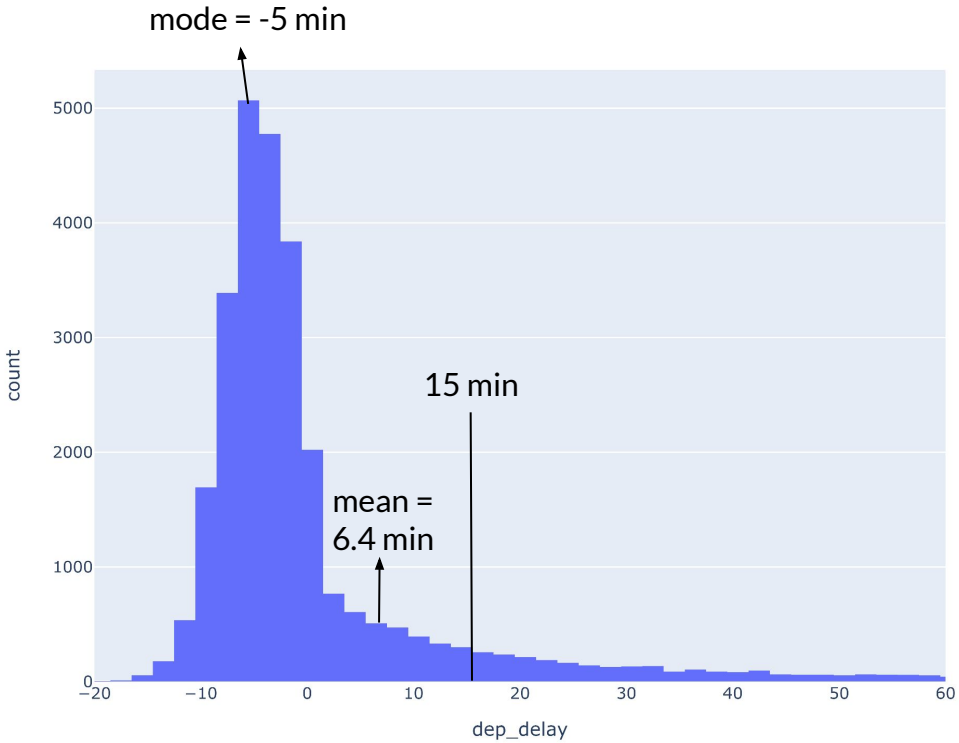
28818 rows × 2 columns

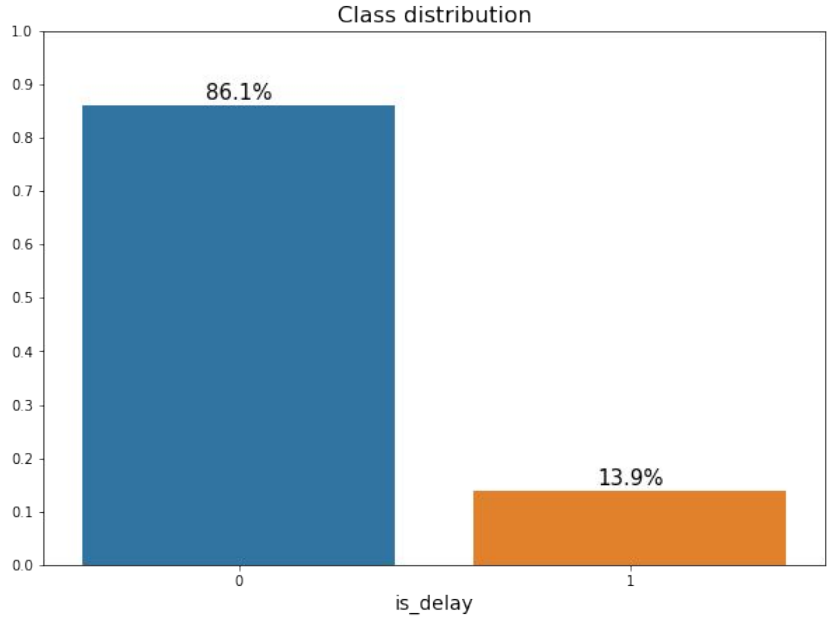
Per tant, és un problema de classificació binària:

{ X: 16 features, y: 1 binary target }

Attribute	Description	Type
month	Month of the year (1-12)	int64
day_of_month	Day of the month (1-31)	int64
day_of_week	Day of the week (1-7)	int64
op_unique_carrier	Airline code (9 total)	object
tail_num	Aircraft registration (plate number) (2092 total)	object
dest	Destination airport IATA code (3 letters) (65 total)	object
dep_delay	Departure delay with respect to scheduled time (min)	int64
crs_elapsed_time	Elapsed time of flight (from take-off to landing) (min)	int64
distance	Flight distance (miles)	int64
crs_dep_m	Scheduled departure time (min of the day)	int64
dep_time_m	Actual departure time (min of the day)	int64
crs_arr_m	Scheduled arrival time (min of the day)	int64
temperature	Airport temperature (ºF)	int64
dew_point	Dew point temperature (ºF)	int64
humidity	Humidity level (0-100%)	int64
wind	Wind direction (16 cardinal directions + CALM + VAR) (18 total)	object
wind_speed	Wind speed (mph)	int64
wind_gust	Wind gust (brief increase in the speed of the wind) (mph)	int64
pressure	Atmospheric pressure (inHg)	float64
condition	Sky condition (clouds, rain, fog, snow, etc.) (25 total)	object
sch_dep	Number of flights scheduled for departure	int64
sch_arr	Number of flights scheduled for arrival	int64
taxi_out	Taxi out time (min)	int64

An imbalanced dataset





Exploratory Data Analysis (EDA)

- Dataset molt net, només 2 files amb NaNs ⟶ **ELIMINADES** 💀
- Drop de la columna ‘tail_num’ (matrícula de l’aeronau)
- Per mostrar els resultats de l’EDA, s’ha fet un *dashboard*:

https://jfkdepartures.herokuapp.com

Comparació dels classificadors

```
models = {
            "KNN" : KNeighborsClassifier(),
            "SVM" : svm.SVC(),
            "Decision Tree" : DecisionTreeClassifier(),
            "Random Forest" : RandomForestClassifier(),
            "Gaussian N-B" : GaussianNB(),
            "Logistic Regression" : LogisticRegression(),
            "Gradient Boosting" : GradientBoostingClassifier()
        }
```

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Recall = \frac{TP}{TP + FN}$$

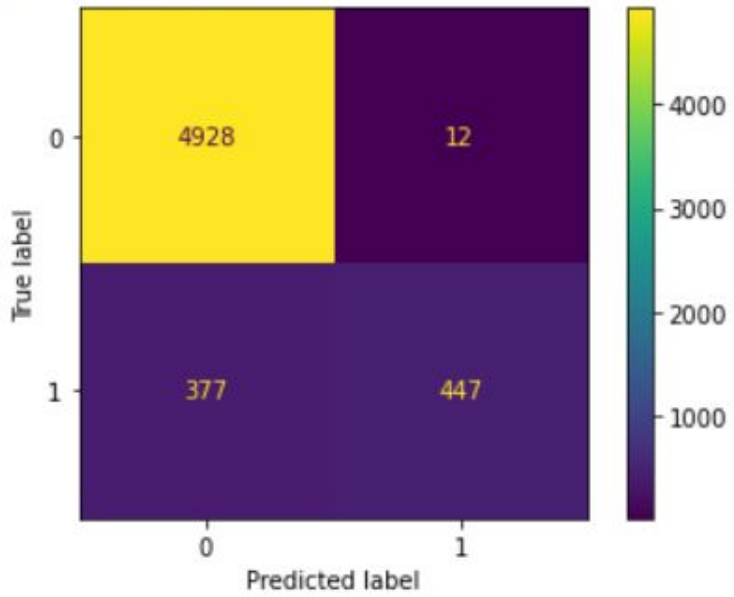
$$Precision = \frac{TP}{TP + FP}$$

$$F1\text{-}score = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

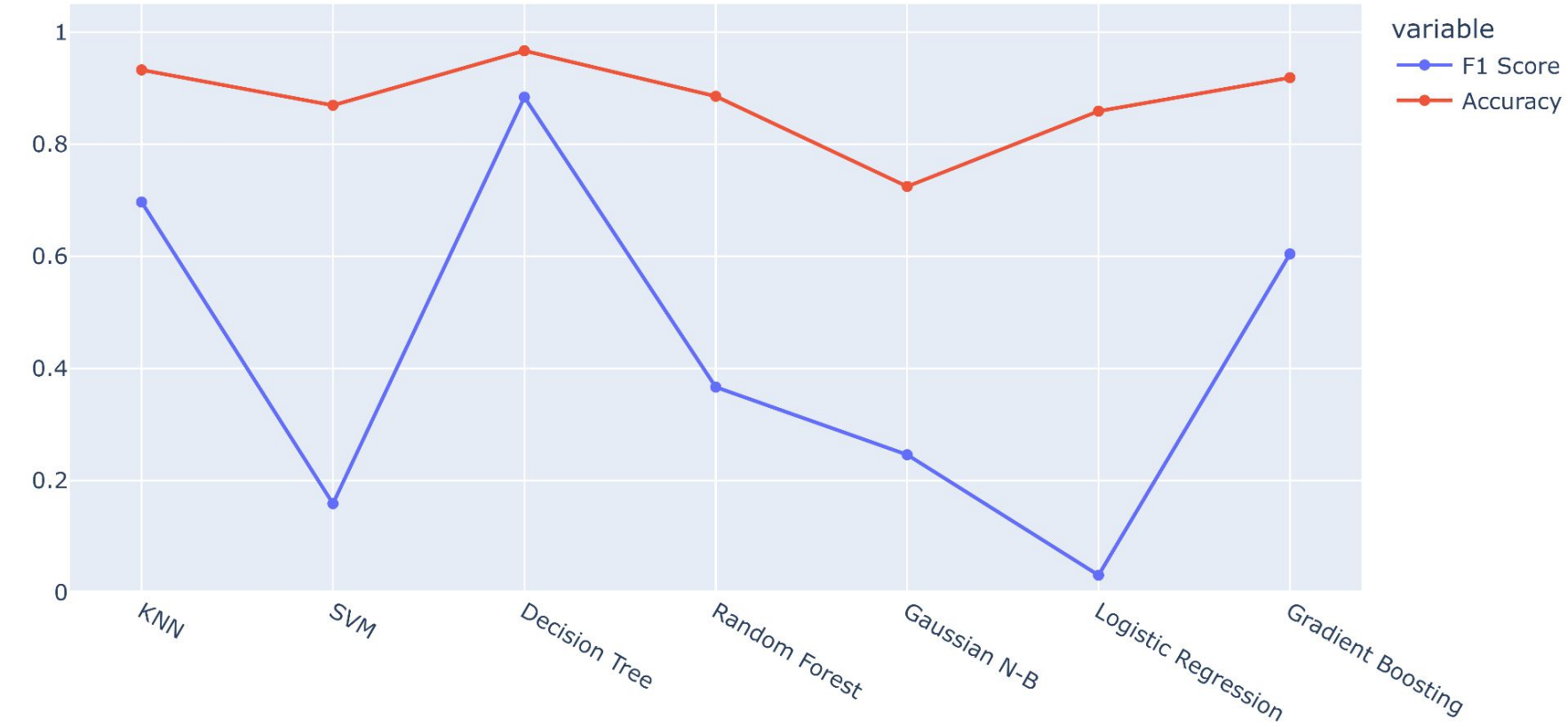
```python
def get_results(model_name, model, X_train, X_test, y_train, y_test):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    # Get the confusion matrix
    cm = confusion_matrix(y_test, y_pred)
    print(f"{model_name} results:")
    cm_display = ConfusionMatrixDisplay(cm).plot()
    plt.show()

    # Get the scores
    acc = accuracy_score(y_test, y_pred)
    pre = precision_score(y_test, y_pred)
    rec = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)

    # Print the results
    print(f"- Accuracy score: {acc:.4f}")
    print(f"- Precision score: {pre:.4f}")
    print(f"- Recall score: {rec:.4f}")
    print(f"- F1 score: {f1:.4f}")
    print()

    return acc, pre, rec, f1
```

KNN results:



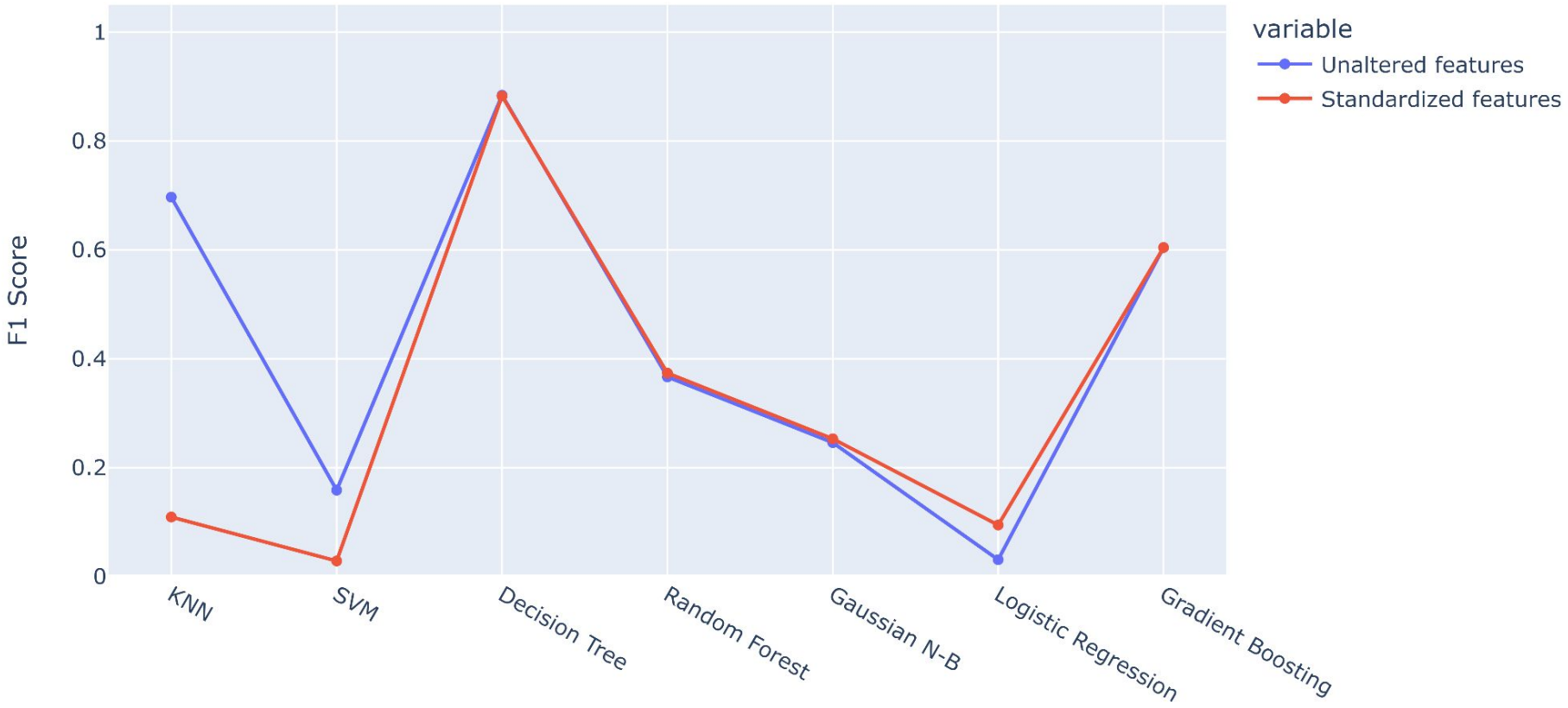
- Accuracy score: 0.9325
- Precision score: 0.9739
- Recall score: 0.5425
- F1 score: 0.6968

Scores comparison for each classifier



F1 Score comparison: unaltered vs standardized features

```
X_norm = StandardScaler().fit_transform(X)
```



Synthetic Minority Oversampling Technique (SMOTE)

```
Counter({0: 19858, 1: 3196})
Class 0 elements: 86.14 %
Class 1 elements: 13.86 %
```

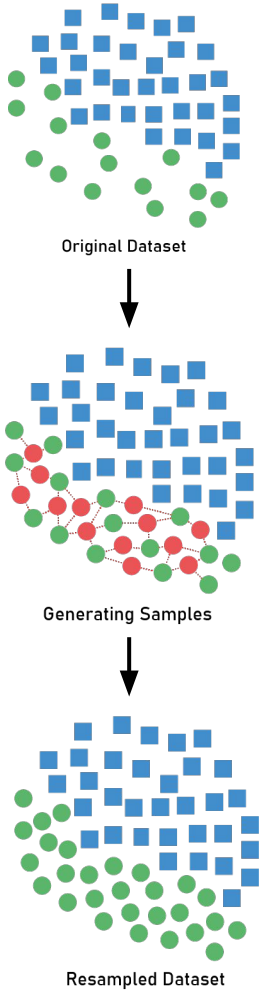
```
from imblearn.over_sampling import SMOTE
```

```
oversample = SMOTE(random_state=42)
```

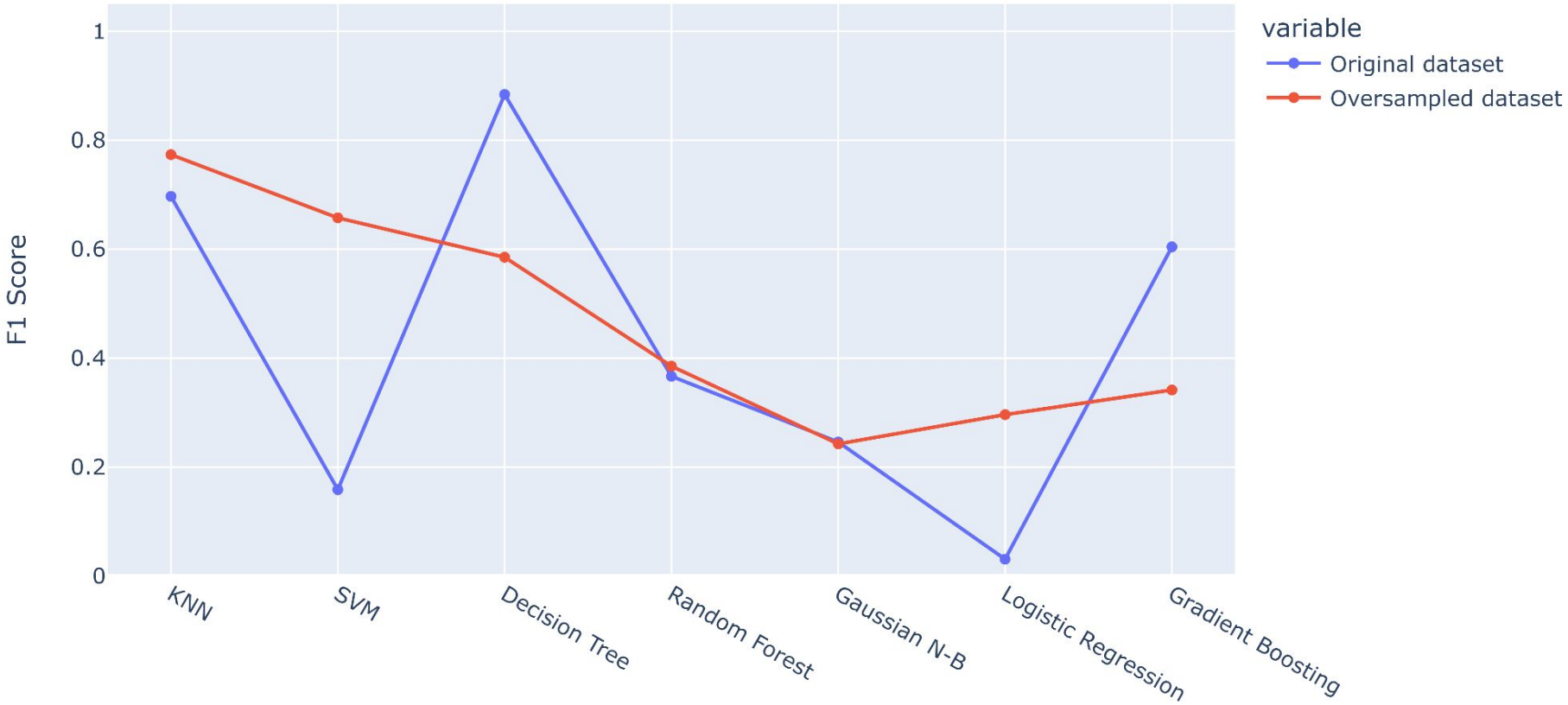
```
X_train_over, y_train_over = oversample.fit_resample(X_train, y_train)
```

```
counter = Counter(y_train_over)
print(counter)
print(f"Class 0 elements: {counter[0]/(counter[0]+counter[1])*100:.2f} %")
print(f"Class 1 elements: {counter[1]/(counter[0]+counter[1])*100:.2f} %")
```

```
Counter({0: 19858, 1: 19858})
Class 0 elements: 50.00 %
Class 1 elements: 50.00 %
```



F1 Score comparison: original vs oversampled dataset

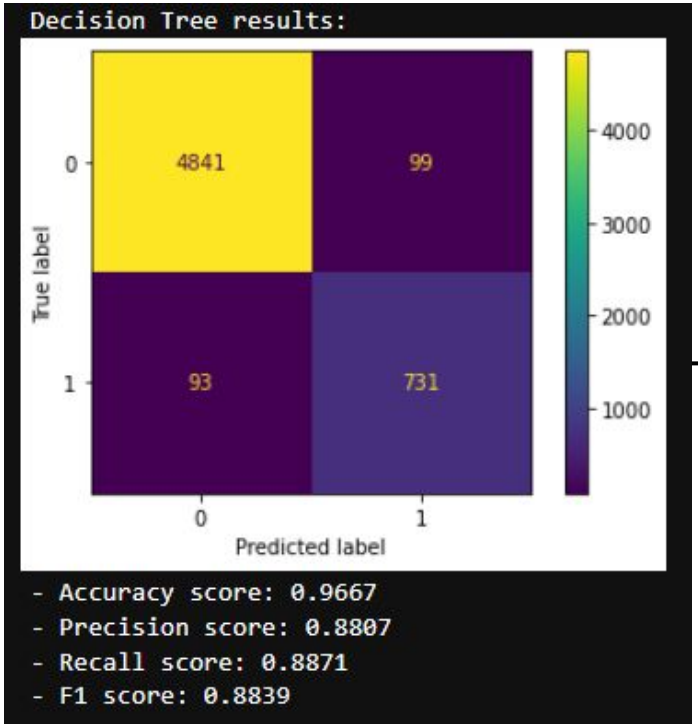


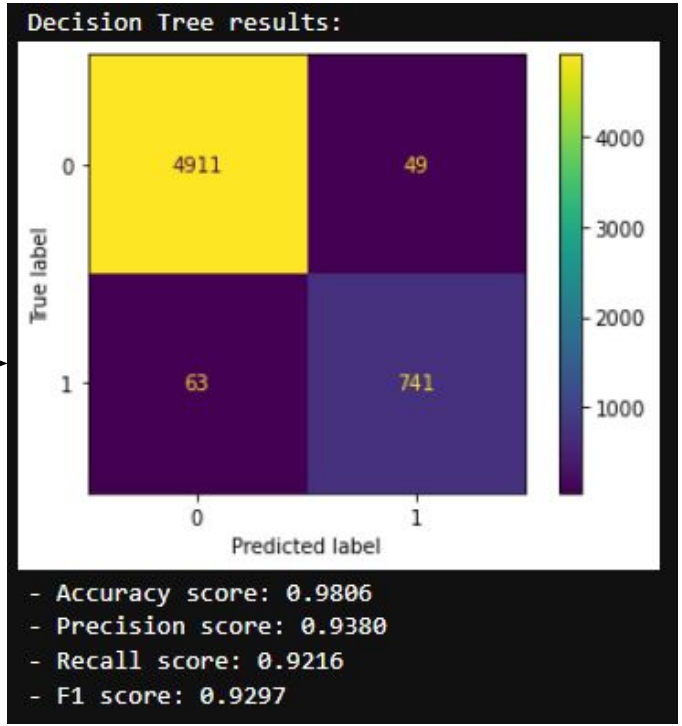
Decision tree tuning

```
train_test_split(X, y, stratify=y, test_size=0.2, random_state=42)
```

```
clf_tree = DecisionTreeClassifier(class_weight='balanced', random_state=42)
```

Original





$$\frac{n_{samples}}{n_{classes} \cdot \text{np.bincount(y)}}$$

Adjust weight inversely prop. to class frequency

Grid Search CV

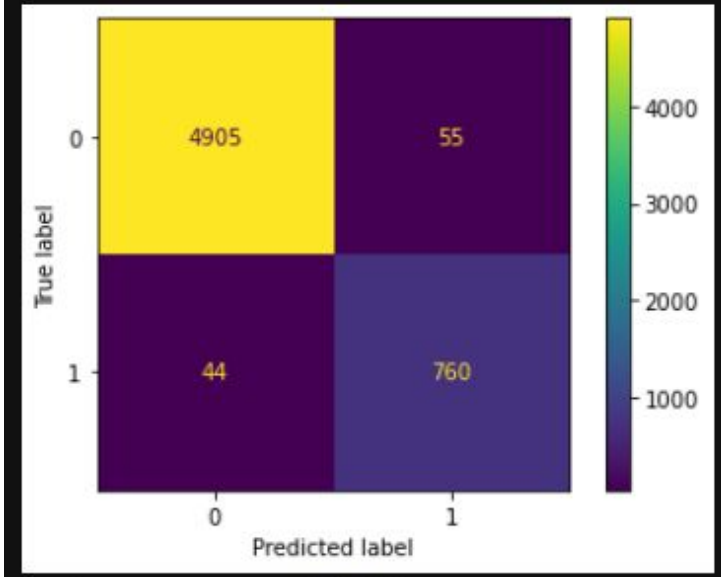
```
max_depth = [25, 26, 27, 28, 29, 30, None]
min_samples_split = [4, 5, 6]
min_samples_leaf = [2, 3]
max_features = ['auto', 'sqrt', None]

param_grid = {'max_depth': max_depth,
              'min_samples_split': min_samples_split,
              'min_samples_leaf': min_samples_leaf,
              'max_features': max_features}

tree_grid = GridSearchCV(estimator = clf_tree,
                         param_grid = param_grid,
                         cv = 5,
                         scoring = ['f1','accuracy','precision','recall'],
                         refit = 'f1',
                         verbose = 3,
                         n_jobs = -1)

tree_grid.fit(X_train, y_train)
```

```
tree_grid.best_params_
```

```
Last executed at 2022-05-12 12:23:04 in 10ms
```

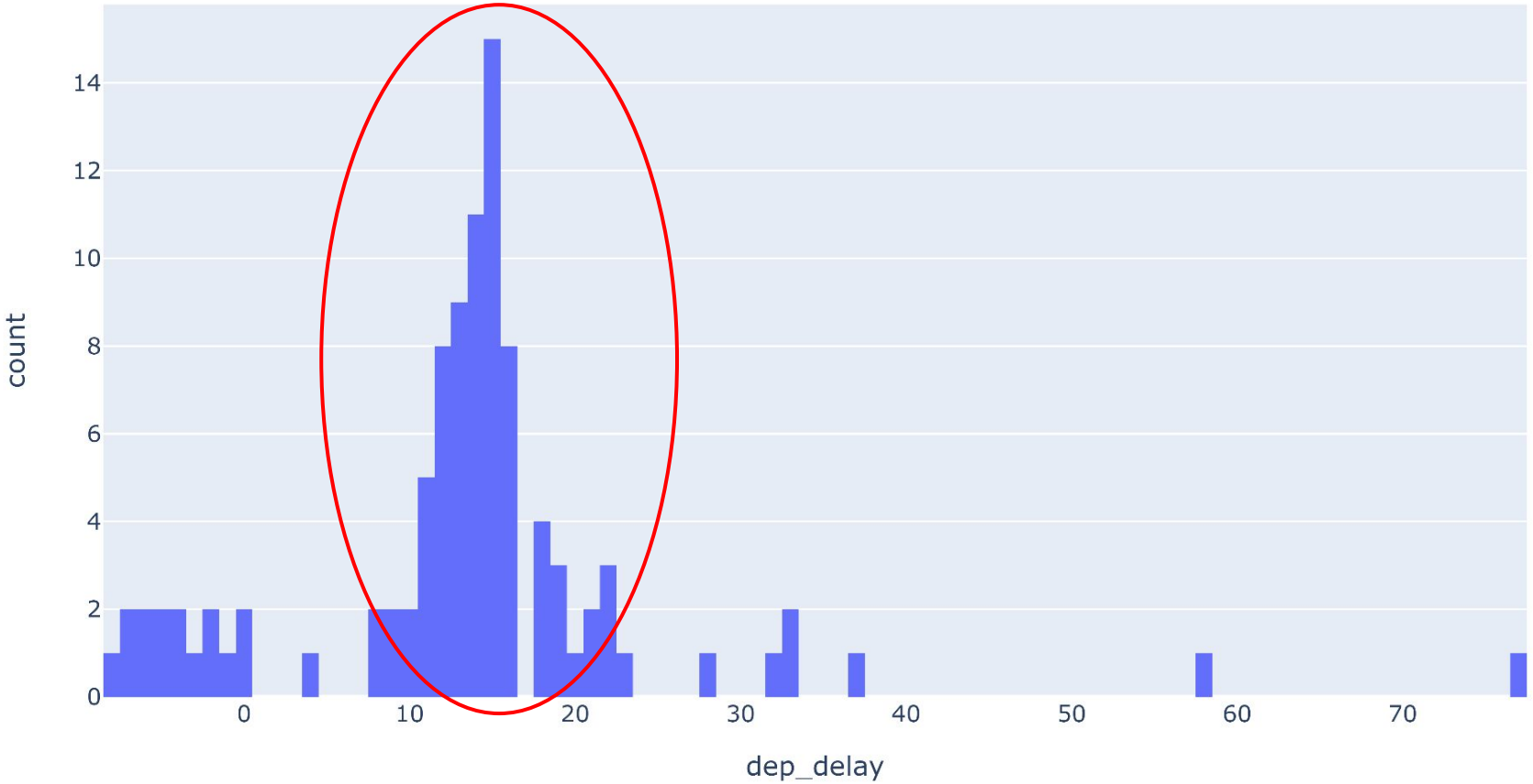
```
{'max_depth': 25,
 'max_features': None,
 'min_samples_leaf': 2,
 'min_samples_split': 5}
```

Best Decision Tree results:



- Accuracy score: 0.9828
- Precision score: 0.9325
- Recall score: 0.9453
- F1 score: 0.9389

Misclassified flights



Conclusions

- Hi ha algoritmes que “toleren” millor els datasets **desbalancejats.**
- El mateix passa quan s’**estandaritzen** les dades i es fa un **oversampling**, per alguns algoritmes és beneficiós i per altres no.
- Per aquest problema en concret, s’han obtingut els millors *scores* fent servir un *Decision Tree*, classificant de forma correcta el **98%** dels vols.

Gràcies per la vostra atenció!

Si teniu algun dubte… 🙄