Predicting the presence of flight arrival delays based on 2015 US air travel data

Vladyslav Lesiv

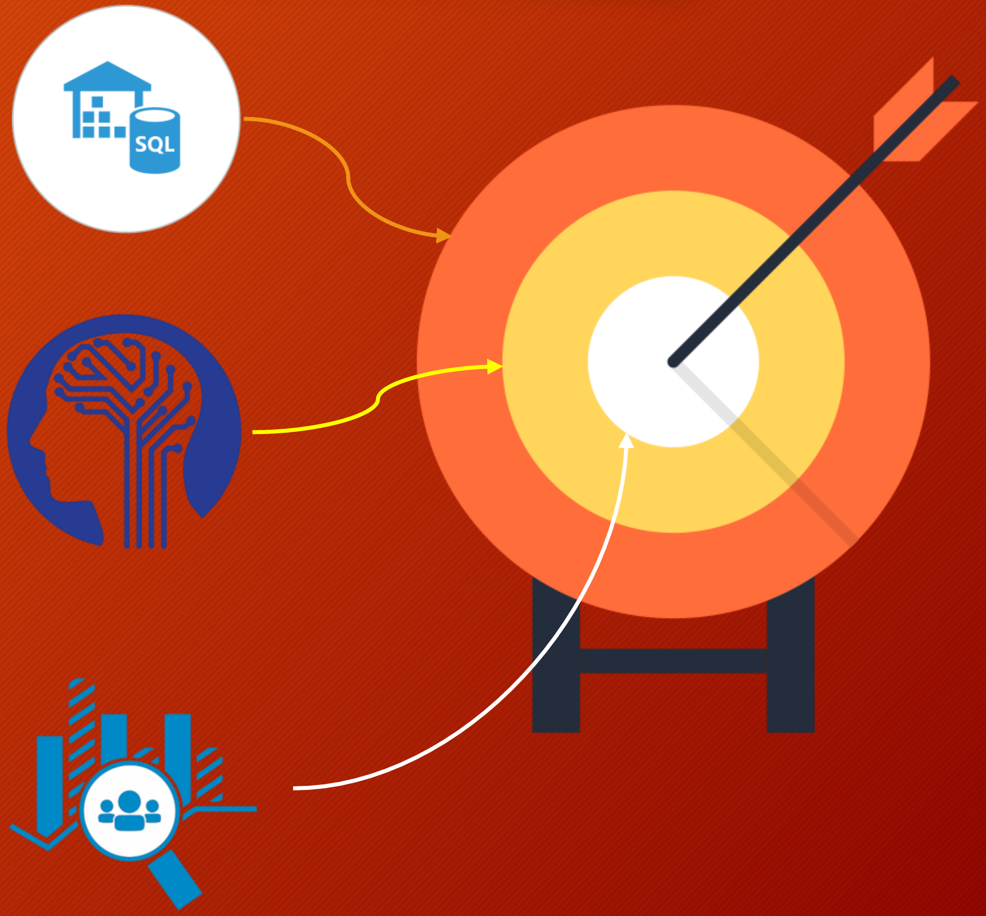
May 21th, 2023

The objective

1. Designing and implementing a data warehouse and ETL processes;

2. Data analysis and selection of methods for research;

3. Evaluation of methods, their comparison, and selection of the best one for predicting airline delays;



Presentation Plan

1. Determining data sources

2. Creating a data warehouse

3. Selecting data analysis and prediction methods

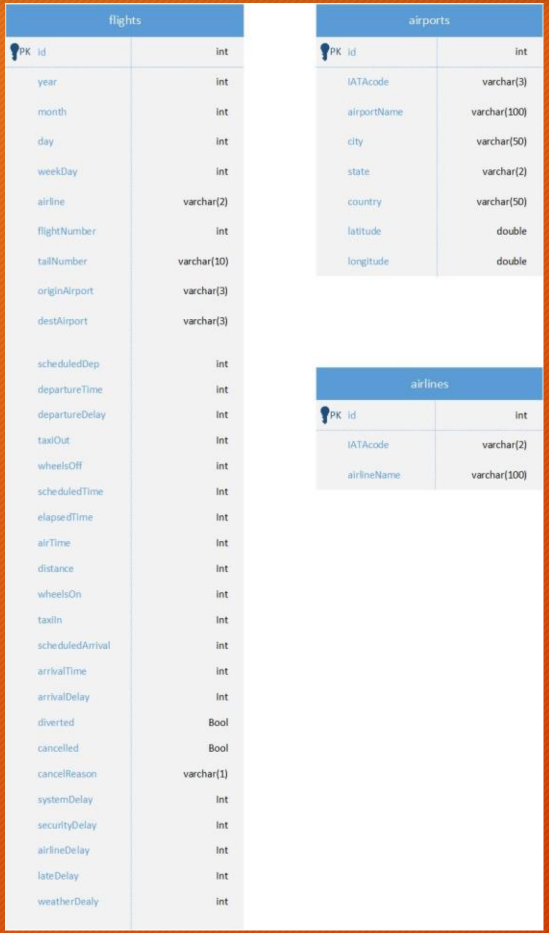
4. Application of methods and their effectiveness comparison

5. Choosing the best forecasting model

The main resources used:

- 2015 Flight Delays and Cancellations. Kaggle: https://www.kaggle.com/datasets/usdot/flight-delays
- Python programming language documentation: https://docs.python.org/3/
- Sklearn library: https://scikit-learn.org/stable/user_guide.html

Creating a data warehouse

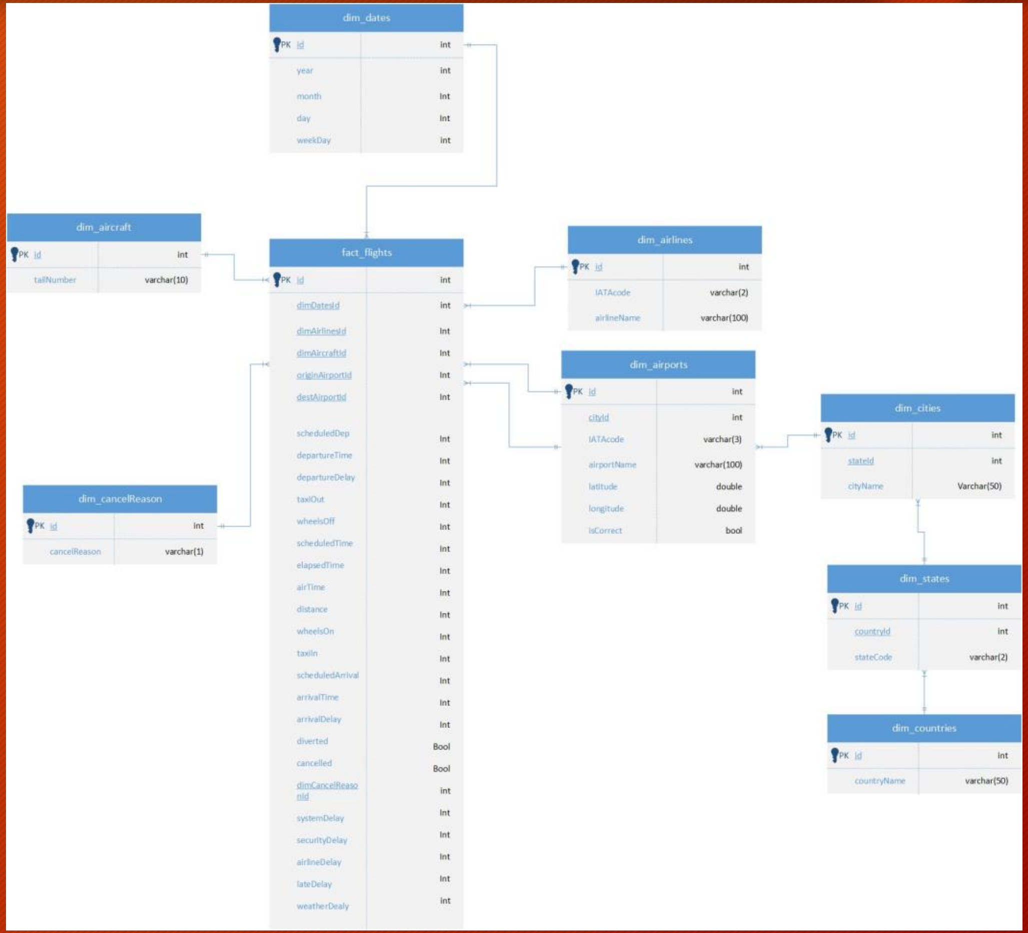


flights		
PK	id	int
	year	int
	month	int
	day	int
	weekDay	int
	airline	varchar(2)
	flightNumber	int
	tailNumber	varchar(10)
	originAirport	varchar(3)
	destAirport	varchar(3)
	scheduledDep	int
	departureTime	int
	departureDelay	Int
	taxiOut	Int
	wheelsOff	int
	scheduledTime	Int
	elapsedTime	Int
	airTime	Int
	distance	Int
	wheelsOn	int
	taxiIn	Int
	scheduledArrival	int
	arrivalTime	int
	arrivalDelay	Int
	diverted	Bool
	cancelled	Bool
	cancelReason	varchar(1)
	systemDelay	Int
	securityDelay	Int
	airlineDelay	Int
	lateDelay	Int
	weatherDealy	int

airports		
PK	id	int
	IATAcode	varchar(3)
	airportName	varchar(100)
	city	varchar(50)
	state	varchar(2)
	country	varchar(50)
	latitude	double
	longitude	double

airlines		
PK	id	int
	IATAcode	varchar(2)
	airlineName	varchar(100)

Initial data format, stage-zone model.



dim_dates		
PK	id	int
	year	int
	month	Int
	day	Int
	weekDay	int

dim_aircraft		
PK	id	int
	tailNumber	varchar(10)

fact_flights		
PK	id	int
	dimDatesId	int
	dimAirlinesId	Int
	dimAircraftId	Int
	originAirportId	Int
	destAirportId	Int
	scheduledDep	Int
	departureTime	Int
	departureDelay	Int
	taxiOut	Int
	wheelsOff	Int
	scheduledTime	Int
	elapsedTime	Int
	airTime	Int
	distance	Int
	wheelsOn	Int
	taxiIn	Int
	scheduledArrival	Int
	arrivalTime	Int
	arrivalDelay	Int
	diverted	Bool
	cancelled	Bool
	dimCancelReasonId	int
	systemDelay	Int
	securityDelay	Int
	airlineDelay	Int
	lateDelay	Int
	weatherDealy	int

dim_cancelReason		
PK	id	int
	cancelReason	varchar(1)

dim_airlines		
PK	id	int
	IATAcode	varchar(2)
	airlineName	varchar(100)

dim_airports		
PK	id	int
	cityId	int
	IATAcode	varchar(3)
	airportName	varchar(100)
	latitude	double
	longitude	double
	isCorrect	bool

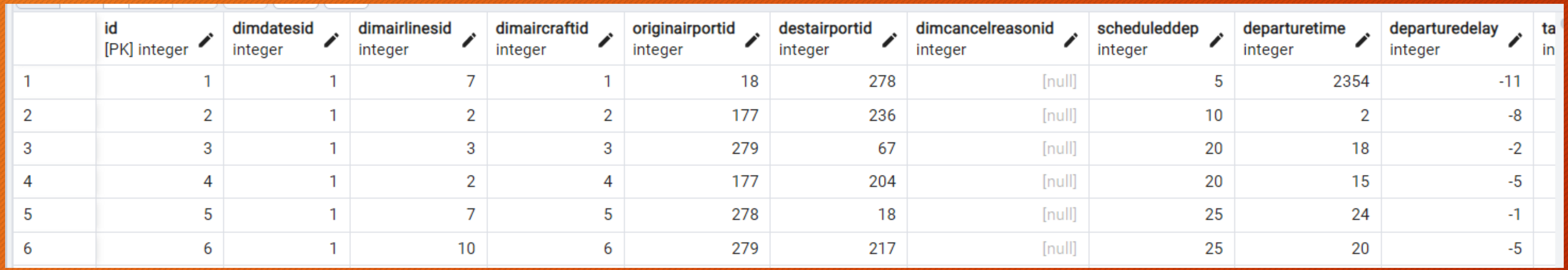
dim_cities		
PK	id	int
	stateId	int
	cityName	Varchar(50)

dim_states		
PK	id	int
	countryId	int
	stateCode	varchar(2)

dim_countries		
PK	id	int
	countryName	varchar(50)

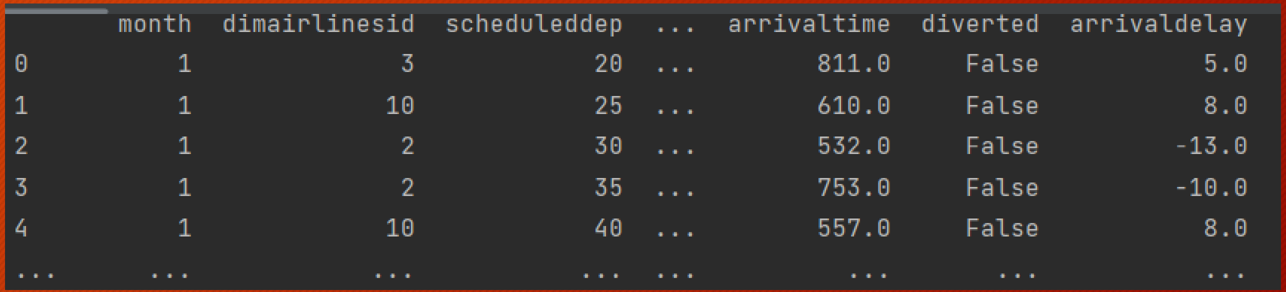
Format of the processed data, data warehouse model.

Data processing



	id [PK] integer	dimdatesid integer	dimairlinesid integer	dimaircraftid integer	originairportid integer	destairportid integer	dimcancelreasonid integer	scheduleddep integer	departuretime integer	departuredelay integer	ta in
1	1	1	7	1	18	278	[null]	5	2354	-11	
2	2	1	2	2	177	236	[null]	10	2	-8	
3	3	1	3	3	279	67	[null]	20	18	-2	
4	4	1	2	4	177	204	[null]	20	15	-5	
5	5	1	7	5	278	18	[null]	25	24	-1	
6	6	1	10	6	279	217	[null]	25	20	-5	

Data from the fact table that will be used for analysis.



```
     month  dimairlinesid  scheduleddep  ...  arrivaltime  diverted  arrivaldelay
0        1              3            20  ...        811.0     False           5.0
1        1             10            25  ...        610.0     False           8.0
2        1              2            30  ...        532.0     False         -13.0
3        1              2            35  ...        753.0     False         -10.0
4        1             10            40  ...        557.0     False           8.0
...    ...            ...           ...  ...          ...       ...           ...
```

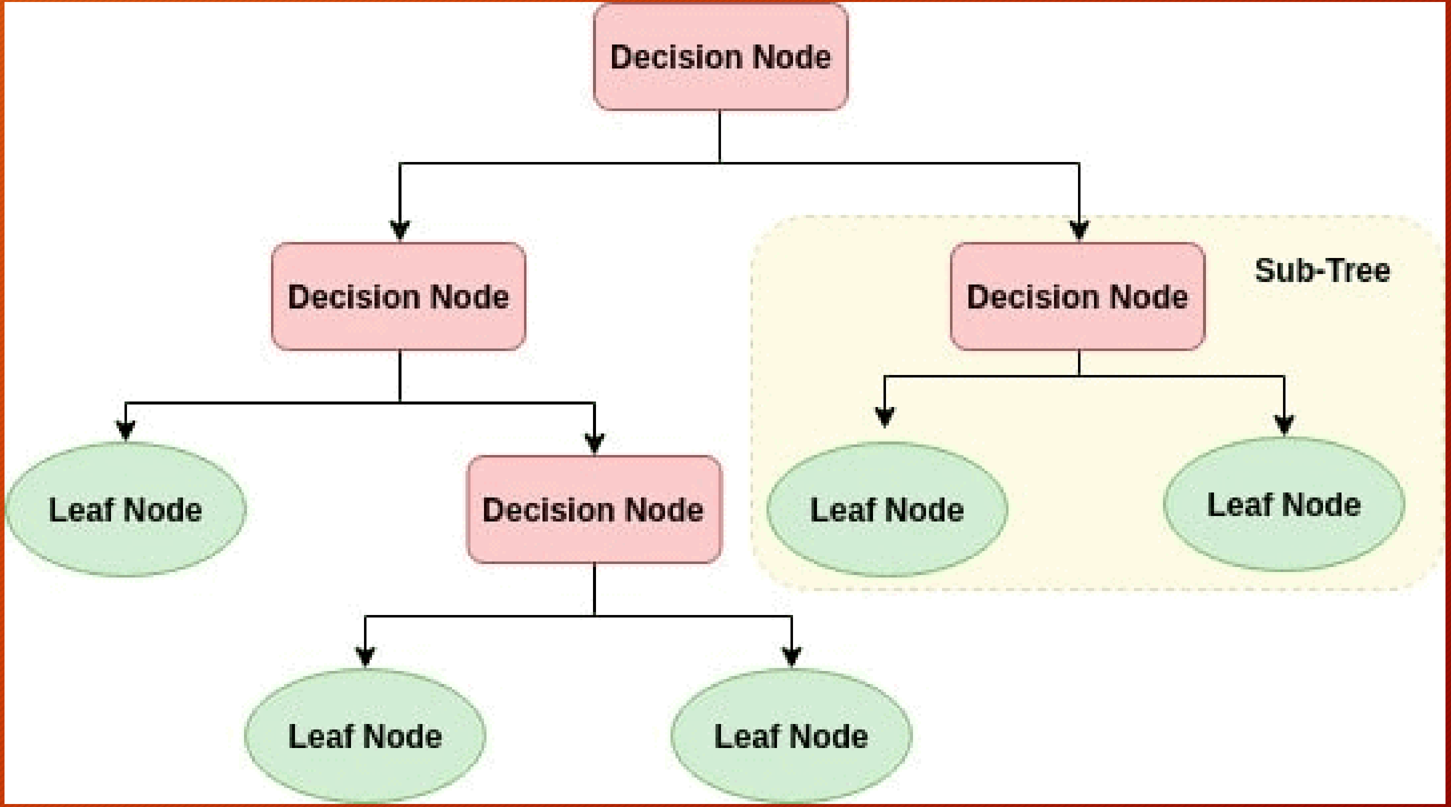
The same data is loaded into the Dataframe with selected factors and corrected errors.

Justification of the selected analysis methods

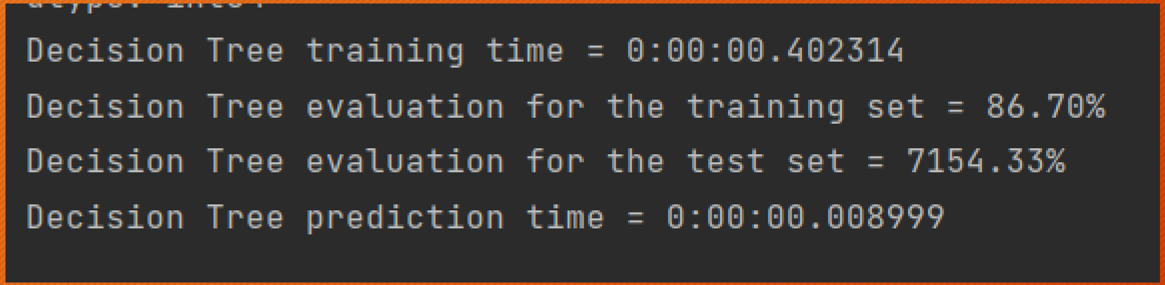
Decision Tree Classifier	K-Nearest Neighbors	SVC (Support Vector Machine)	K-Means
Simple, easy to interpret	The lazy learning algorithm	Works efficiently with data with many features	Algorithm of learning without a teacher
Has a high performance	Easy to understand and implement	High classification accuracy	Has fast convergence

Model 1 – Decision Tree Classifier

Decision Tree Classifier creates a classification model by building a decision tree. Each node in the tree defines a test for an attribute, and each branch descending from this node corresponds to one of the possible values of the attribute.

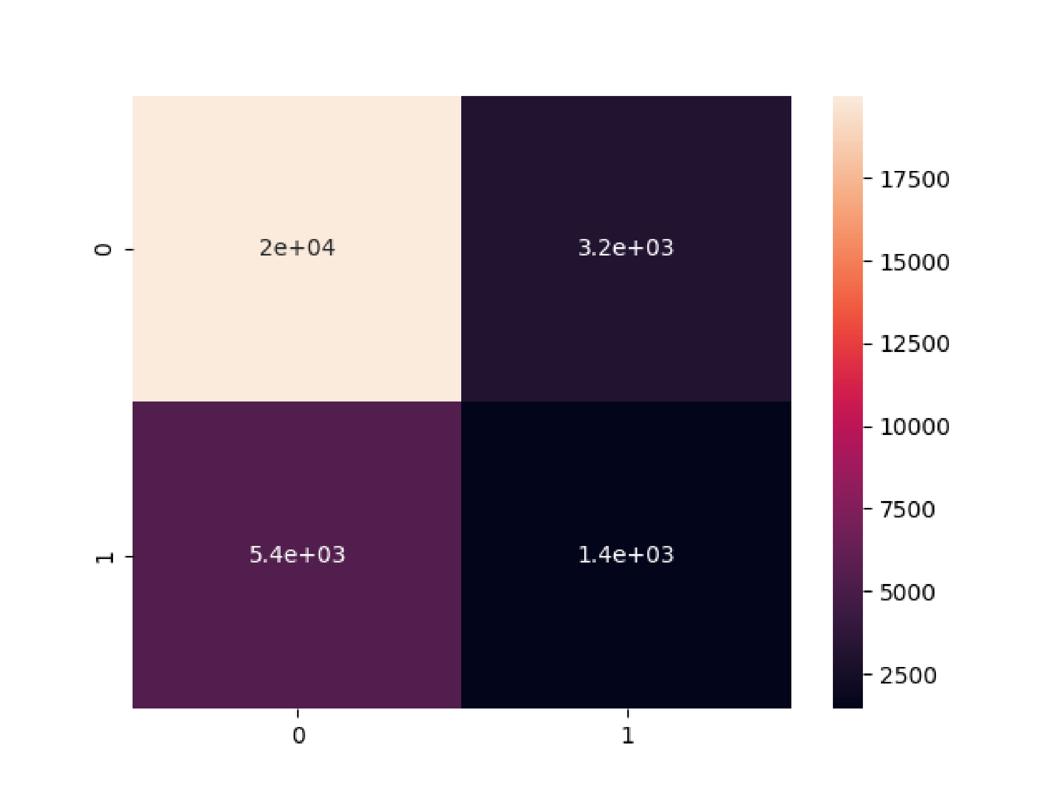


Model 1 – Results and implications.



```
dtype: int64
Decision Tree training time = 0:00:00.402314
Decision Tree evaluation for the training set = 86.70%
Decision Tree evaluation for the test set = 7154.33%
Decision Tree prediction time = 0:00:00.008999
```

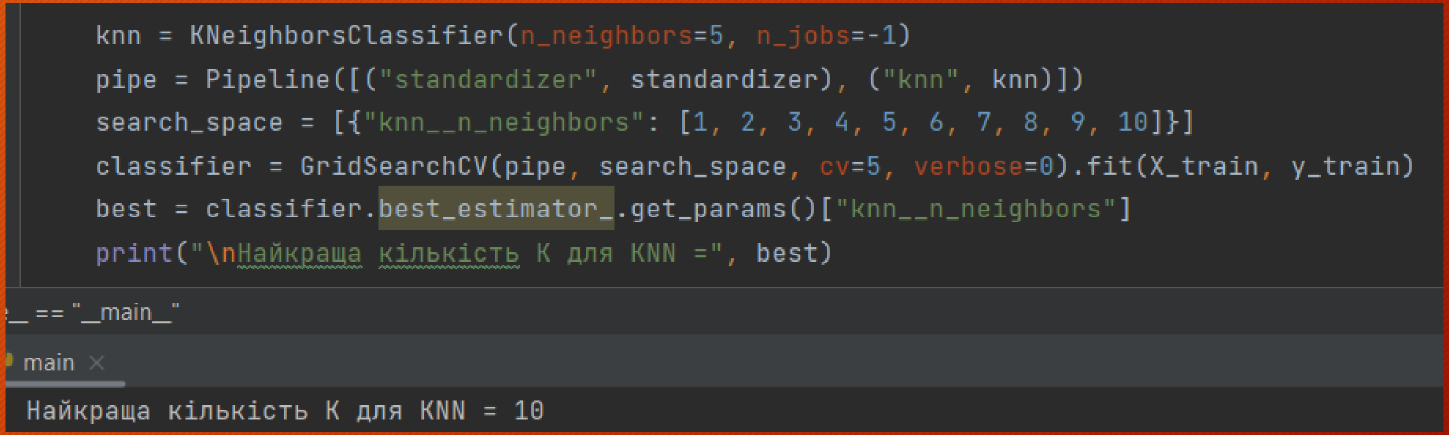
We can see that Decision Tree quickly coped with the classification task, the time is measured in milliseconds, and the prediction scores are quite high. The score for the training set is 86.70%, and the test set is 71.54%. We can state that the model works effectively.



Inconsistency matrix for Decision Tree Classifier.

Model 2 – K-Nearest Neighbors

KNN uses a set of data examples with known class labels and classifies new examples by comparing them to their nearest neighbors. The "K" in KNN indicates the number of nearest neighbors that the algorithm uses to solve the classification.



```
    knn = KNeighborsClassifier(n_neighbors=5, n_jobs=-1)
    pipe = Pipeline([("standardizer", standardizer), ("knn", knn)])
    search_space = [{"knn__n_neighbors": [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]}]
    classifier = GridSearchCV(pipe, search_space, cv=5, verbose=0).fit(X_train, y_train)
    best = classifier.best_estimator_.get_params()["knn__n_neighbors"]
    print("\nНайкраща кількість K для KNN =", best)
_ == "_main_"
```

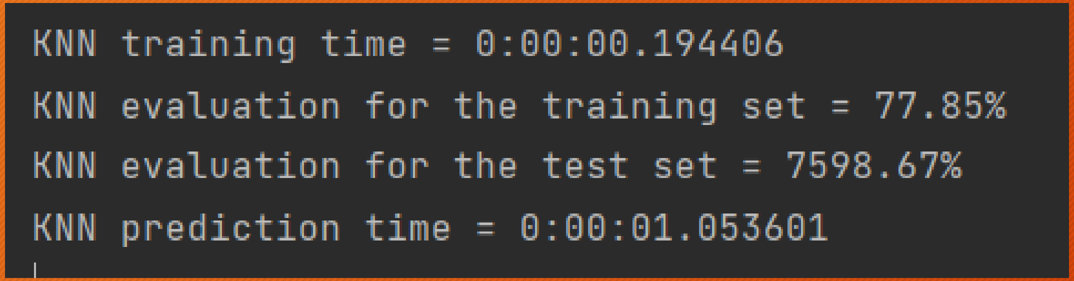
```
Найкраща кількість K для KNN = 10
```

Finding the best K.

$$d(c, q) = \sqrt[2]{(q_1 - c_1)^2 + (q_2 - c_2)^2}$$

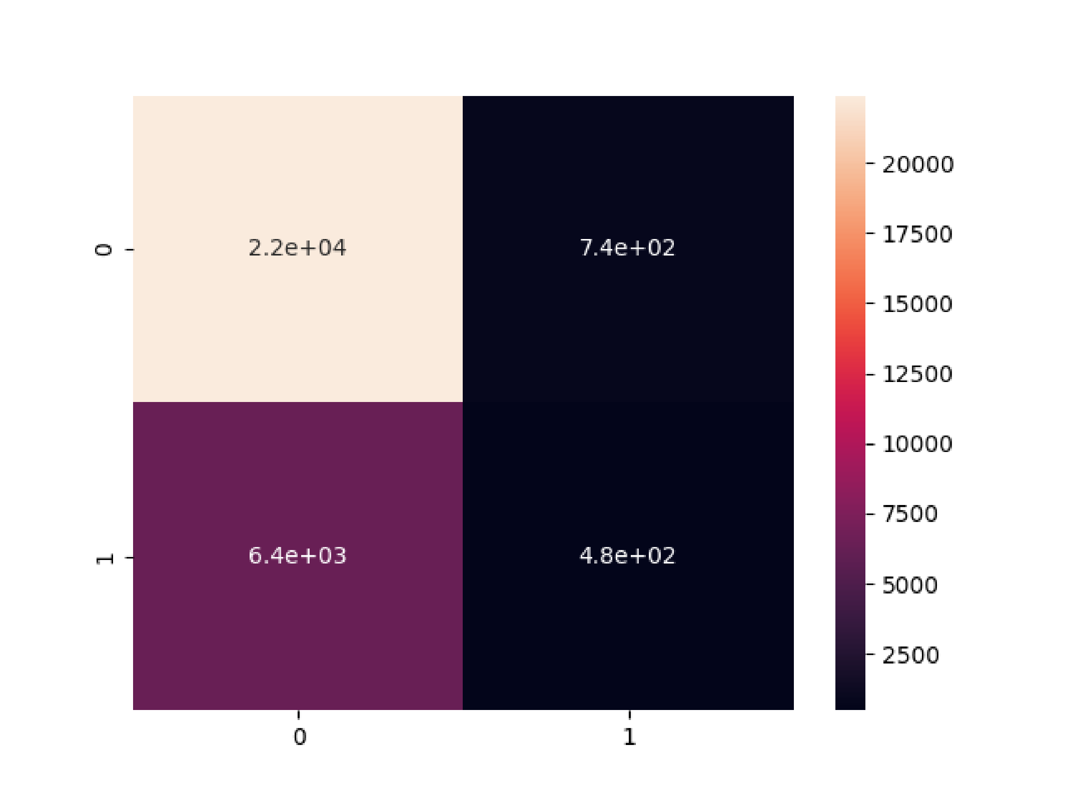
The metric used is the Euclidean distance.

Model 2 – Results and implications.



```
KNN training time = 0:00:00.194406
KNN evaluation for the training set = 77.85%
KNN evaluation for the test set = 7598.67%
KNN prediction time = 0:00:01.053601
```

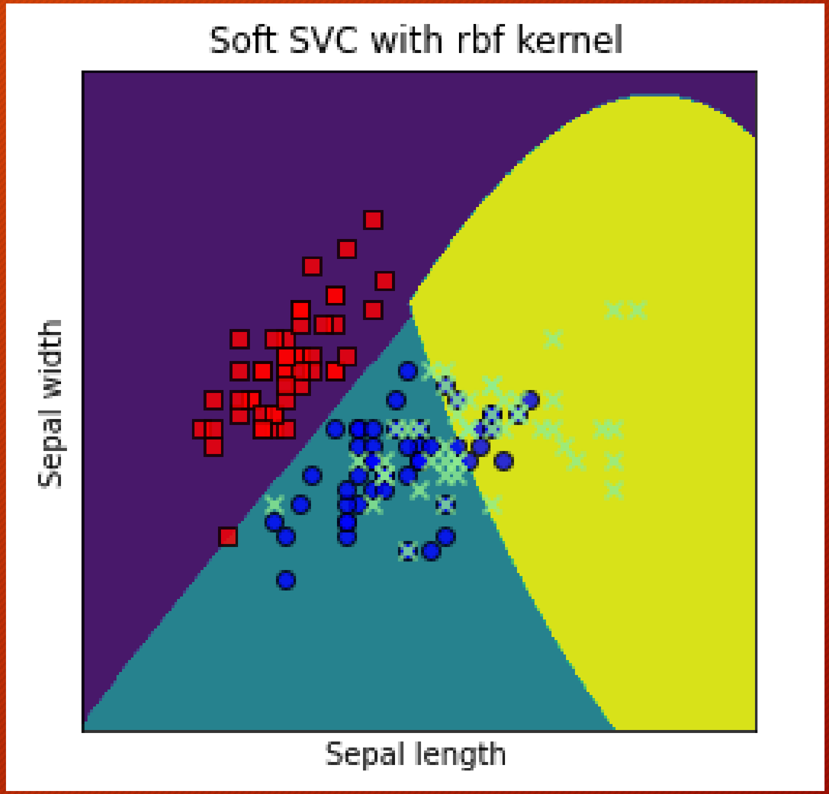
We can see that KNN effectively coped with the classification task. The training set score is 77.85%, which is lower than that of Decision Tree, and the test set score is 75.99%, which is higher. Since the score of the test set is more important, we can say that KNN has shown a higher performance in practice.



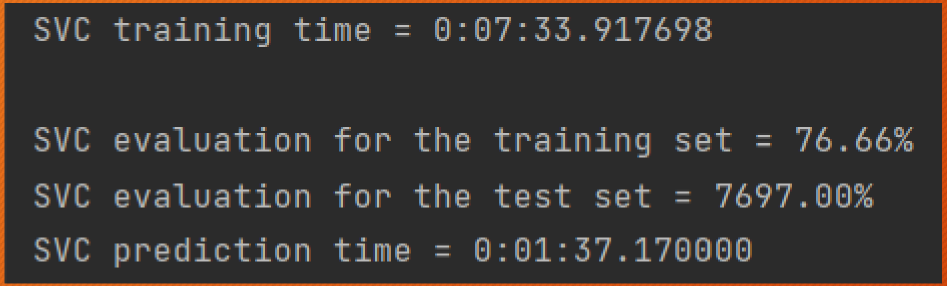
Inconsistency matrix for K-Nearest Neighbors

Model 3 – SVM (Support Vector Machine)

SVM with RBF (Radial Basis Function) kernel uses a nonlinear kernel function to solve nonlinear classification problems. The RBF kernel is used to transform the data into a higher-dimensional space where it becomes linearly separable. It then calculates the distance between the input samples and uses it as a measure of similarity.



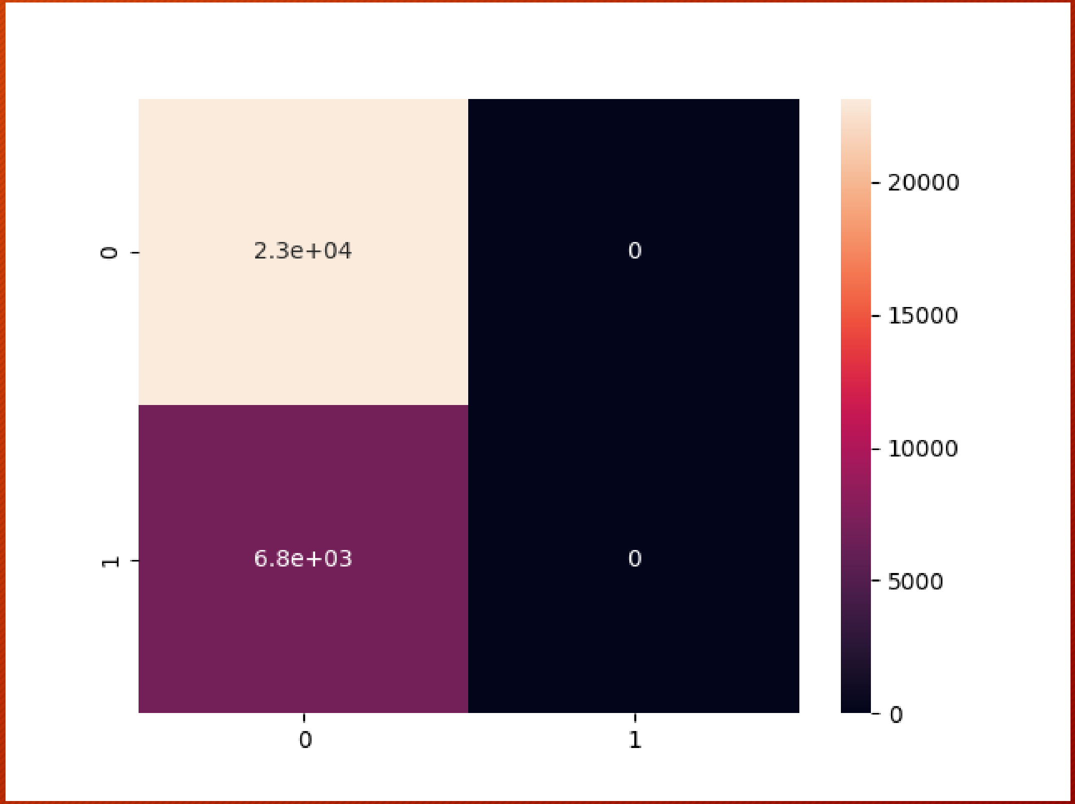
Model 3 – Results and implications.



```
SVC training time = 0:07:33.917698


SVC evaluation for the training set = 76.66%
SVC evaluation for the test set = 7697.00%
SVC prediction time = 0:01:37.170000
```

We can see that SVC is effective in prediction. The training set score is 76.66%, which is about the same as the previous model, and the test set score is 76.97%, which is higher than the rest of the models. However, the training took as long as 7:34 minutes, which is very long, and with larger samples, the training time would be unacceptably long.



Inconsistency matrix for SVC

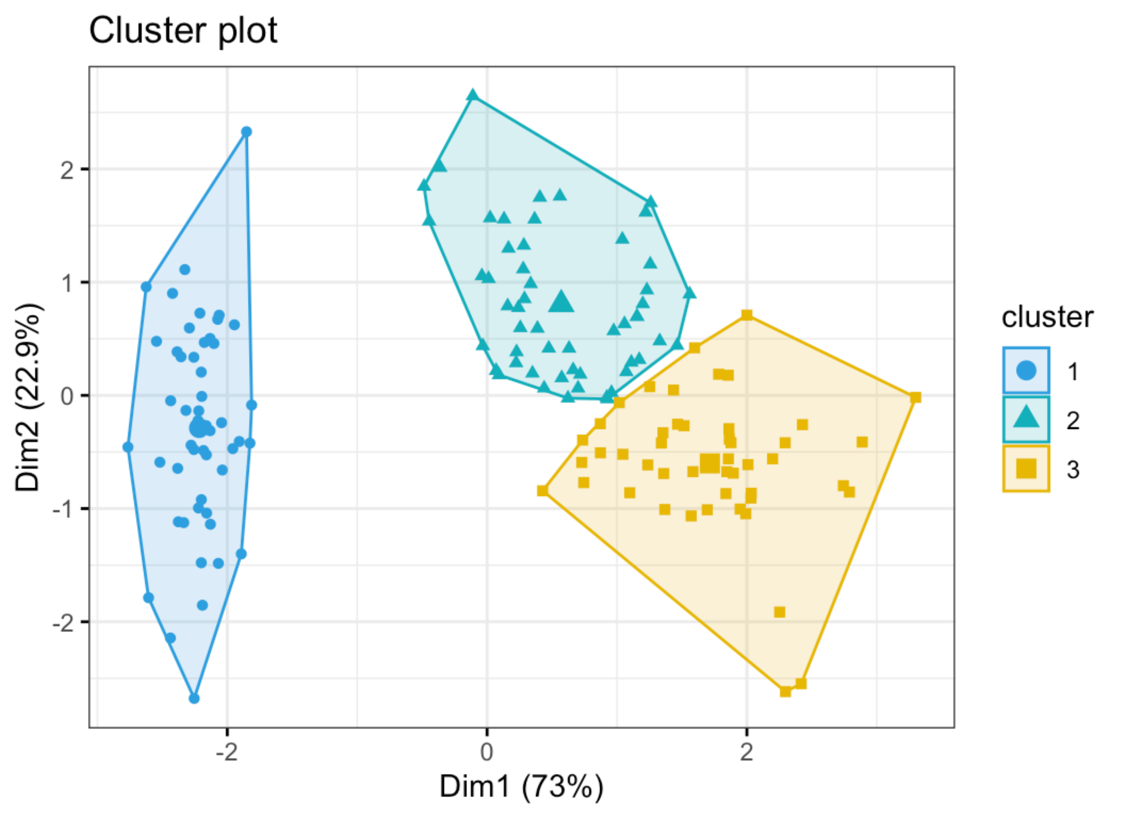
Model 4 – K-Means

The K-Means algorithm is based on the fact that each point in a cluster should be close to the center of that cluster. It works like this: first, we choose k - the number of clusters we want to find in the data. Then we initialize the centers of the clusters - centroids.

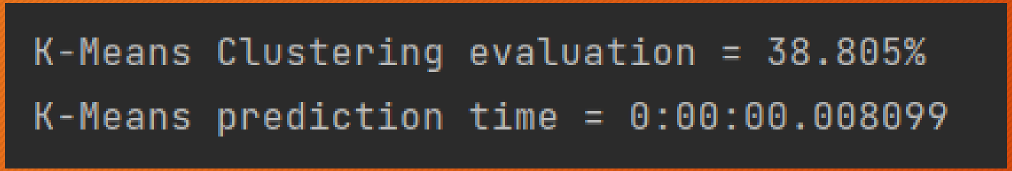
Next, the algorithm performs two parts in turn:

- Assigns each point to the cluster whose centroid is closest to it.
- Recalculates the position of each centroid as the average of all points in the cluster.

The steps are repeated until the centroids stop moving or the points stop moving between clusters.

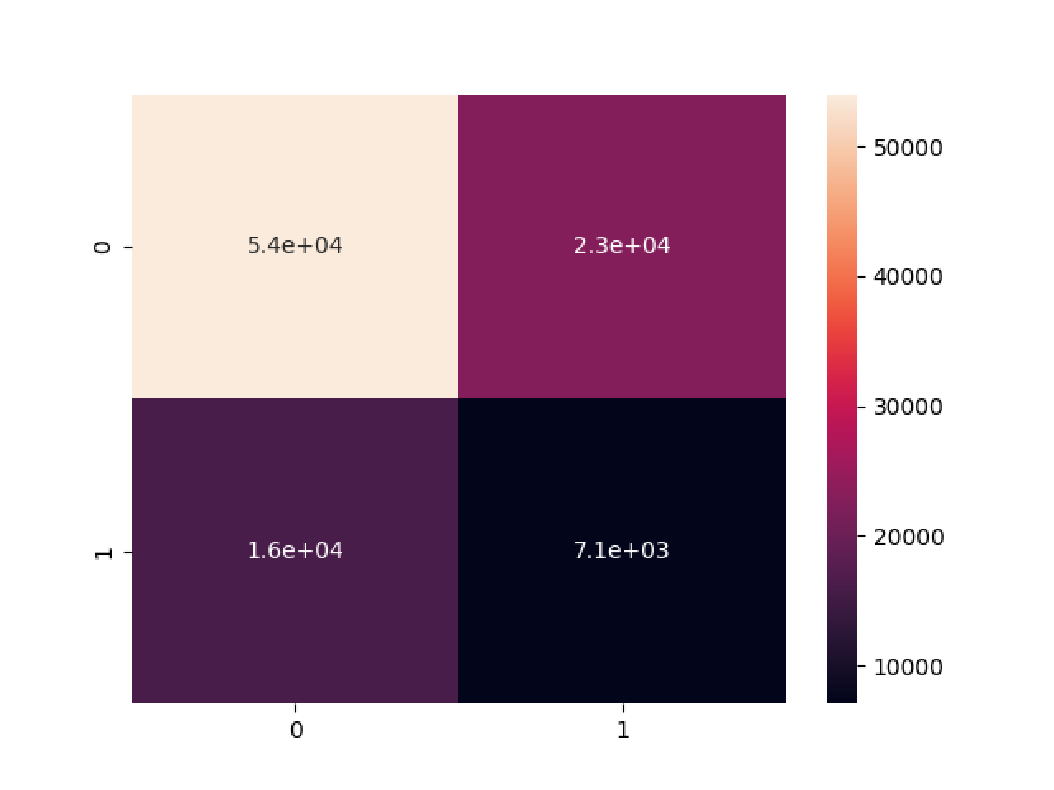


Model 4 – Results and implications.



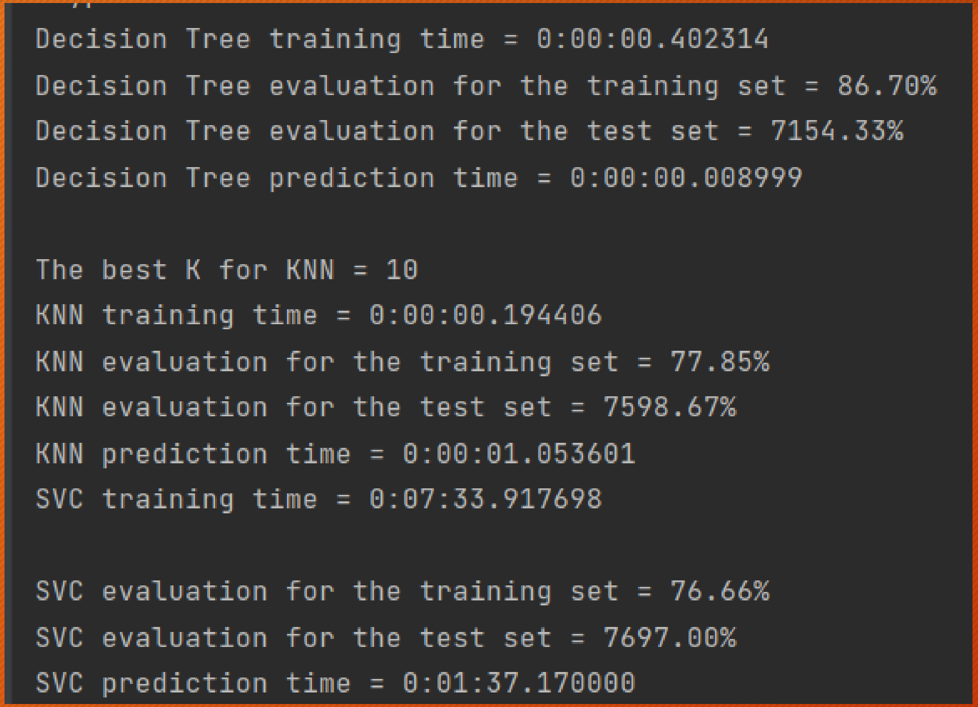
```
K-Means Clustering evaluation = 38.805%
K-Means prediction time = 0:00:00.008099
```

We can see that clustering showed a result of 38.805% of correct predictions. This is a pretty good result for data with this model. The runtime is negligible, so the model works to the best of its ability.



Inconsistency matrix for K-Means

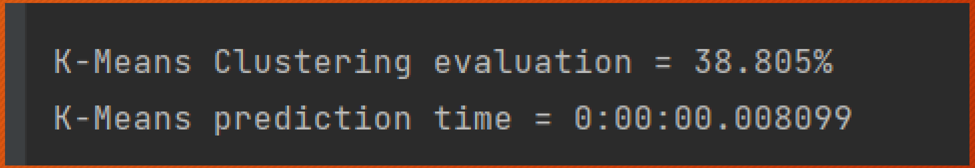
Results



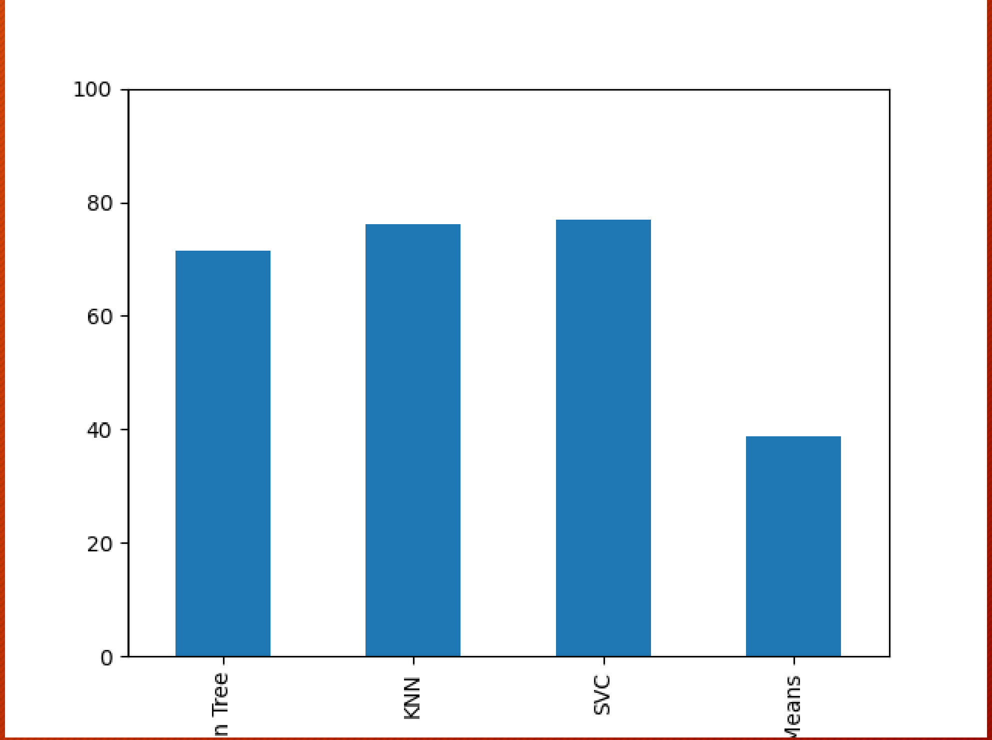
```
Decision Tree training time = 0:00:00.402314
Decision Tree evaluation for the training set = 86.70%
Decision Tree evaluation for the test set = 7154.33%
Decision Tree prediction time = 0:00:00.008999

The best K for KNN = 10
KNN training time = 0:00:00.194406
KNN evaluation for the training set = 77.85%
KNN evaluation for the test set = 7598.67%
KNN prediction time = 0:00:01.053601
SVC training time = 0:07:33.917698

SVC evaluation for the training set = 76.66%
SVC evaluation for the test set = 7697.00%
SVC prediction time = 0:01:37.170000
```



```
K-Means Clustering evaluation = 38.805%
K-Means prediction time = 0:00:00.008099
```



Histogram of prediction results on a test sample.

Conclusions

- Flight delays are a very serious issue that affects the economy, the well-being of individuals and society as a whole. Therefore, it is important to look for and find ways to solve this problem as soon as possible and put them into practice.
- After analyzing four methods, I determined that K-Nearest Neighbors is the best model for predicting flight delays.

Thank you for your attention!