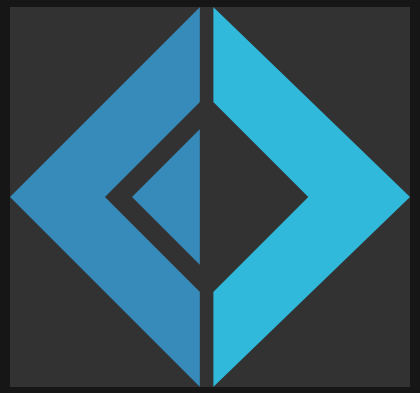
TABLE OF CONTENTS




ML.NET

```
1 F#
```

1.1 UN PEU D'HISTOIRE

(Syme, 2019)

1.1 UN PEU D'HISTOIRE

(Syme, 2019)

Année	Recherche	"Corporate"
1973	ML, le Lisp « typé »	
1975		Microsoft
1995		Java
1997	Standard ML	
1998	Don Syme @ Microsoft Research	
2002		.NET [1]
2005	F#	

1.2 F# ET .NET

- F# est vu comme LE langage fonctionnel par MS sur .NET
- .NET et F# ont co-évolué

1.3 F# ET .NET CORE

- Reboot .NET
- 100% multiplateforme (Mac/Windows/Linux + x86/ARM)
- OSS

1.4 F# VS < X >

1.4 F# VS < X >

AVANTAGES

- Stable
- Cohérent
- Ouvert
- Interopérable

1.4 F# VS < X >

AVANTAGES

- Stable
- Cohérent
- Ouvert
- Interopérable

INCONVÉNIENTS

- Fonctionnalités
- Pas de "killer app"
- Couverture académique

1.5 UTILISATION DANS L'INDUSTRIE

- MS, of course (Cloud et produits internes)
- Jet.com
- Genetec
- Banques et Trading (ingénierie de la finance)
- Développement mobile (Xamarin)

1.6 LES FONCTIONNALITÉS À PART

- Les fournisseurs de type (*Type Providers*) [2]
- Les « motifs actifs » (*Active Patterns*)
- Les « expressions de calcul » (*Computation expressions*)

1.7 FOCUS : LES FOURNISSEURS DE TYPE

Un fournisseur de type F# est un composant qui fournit des types, des propriétés et des méthodes. Les Fournisseurs de type génèrent ce que l'on appelle des *Types fournis*, qui sont générés par le compilateur de `F#` et sont basés sur une source de données externe.

Par exemple, un fournisseur de Type pour SQL peut générer des types représentant les tables et colonnes dans une base de données relationnelle. En fait, c'est ce que fait `SQLProvider`.

Les types fournis dépendent dépendent des paramètres d’entrée d'un fournisseur de Type. Ces paramètres peuvent être un échantillon de source de données (par exemple, un fichier JSON), une URL qui pointe directement vers un service externe ou une chaîne de connexion à une source de données.

Un fournisseur de Type peut également vous garantir que les groupes de types sont "instanciés" uniquement à la demande ; Autrement dit, ils ne sont appelés que si les types sont réellement référencés par votre programme. Cela permet l'intégration directe et à la demande d'espaces de données à grande échelle, tels que les marchés de données (*data market*) en ligne, de manière fortement typée.

1.8 TYPE PROVIDER I : OPEN DATA

In [4]:

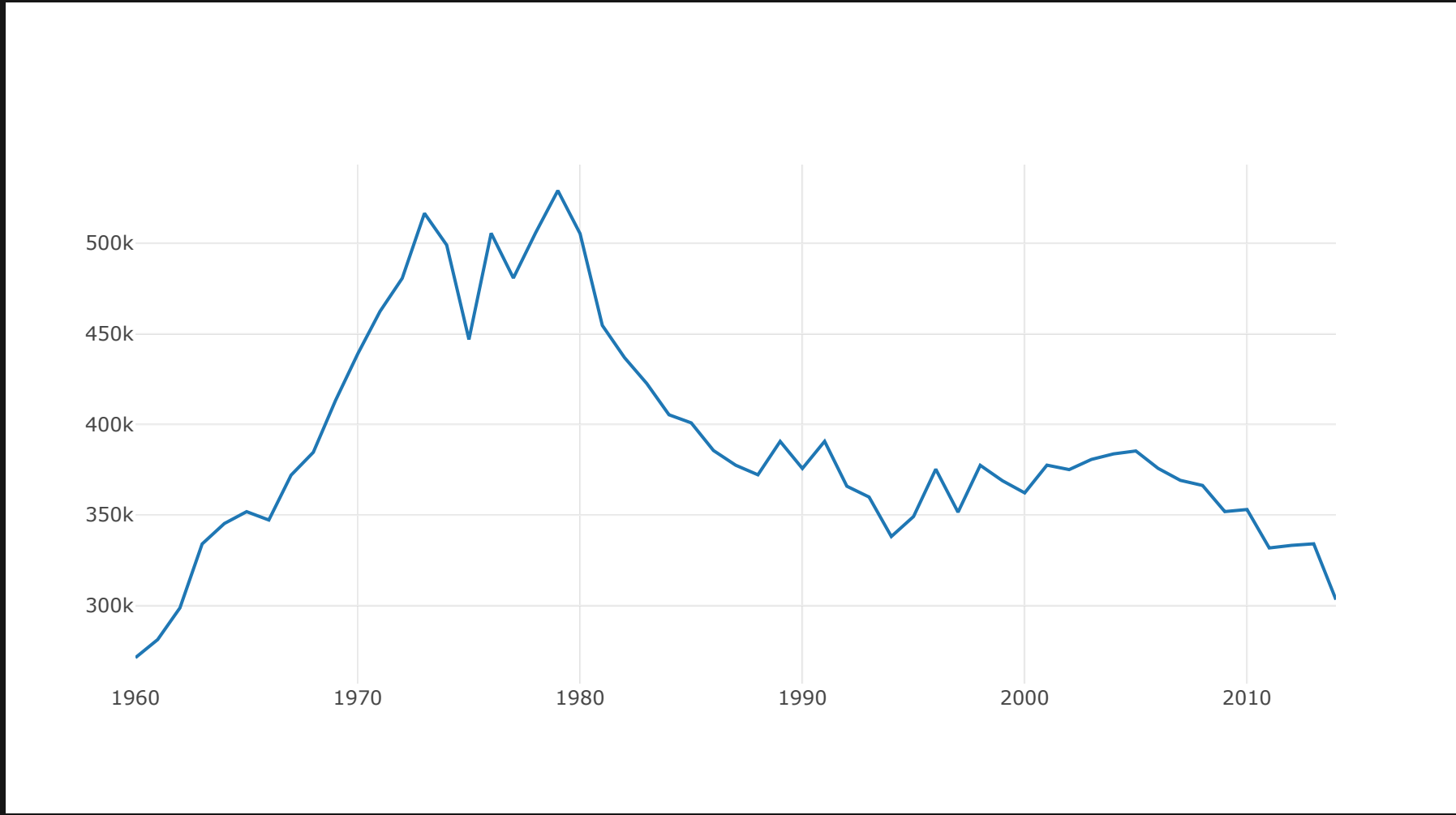
```
open FSharp.Data
open XPlot.Plotly

let wb = WorldBankData.GetDataContext()
wb
```

Out[4]: FSharp.Data.Runtime.WorldBank.WorldBankData

In [5]:

```
// Obtenir les émissions de CO2 en kt par an pour la France, l'Allemagne et les
wb.Countries.France.Indicators.``CO2 emissions (kt)``
|> Chart.Line
```

Out[5]:



1.9 TYPE PROVIDER II : SQL

In [6]:

```
[<Literal>]
let ConnectionString =
    "Data Source=" +
    __SOURCE_DIRECTORY__ + @"/data/northwindEF.db;" +
    "Version=3;foreign keys=true"

[<Literal>]
let ResolutionPath = __SOURCE_DIRECTORY__ + @"/local"

open FSharp.Data.Sql

type Sql = SqlDataProvider<
                Common.DatabaseProviderTypes.SQLITE,
                SQLiteLibrary = Common.SQLiteLibrary.SystemDataSQLite,
                ConnectionString = ConnectionString,
                ResolutionPath = ResolutionPath,
                CaseSensitivityChange = Common.CaseSensitivityChange.ORIGINAL>

let db = Sql.GetDataContext()
```

In [7]:

```
// liste des noms des clients
db.Main.Customers
|> Seq.map (fun c -> c.ContactName)
```

Out[7]:

```
seq ["Maria Anders"; "Ana Trujillo"; "Antonio Moreno"; "Thomas Hardy";
...]
```

2 ML.NET

[3]

GÉNÉRALITÉS

GÉNÉRALITÉS

- 10 années de travaux internes, collaboration Microsoft Research et composantes appliquées (logiciels)

GÉNÉRALITÉS

- 10 années de travaux internes, collaboration Microsoft Research et composantes appliquées (logiciels)

- Pensé dès le départ pour le machine learning

GÉNÉRALITÉS

- 10 années de travaux internes, collaboration Microsoft Research et composantes appliquées (logiciels)
- Pensé dès le départ pour le machine learning
- Représentation des données (*DataView*) fonctionnelle : immutable, à la demande, inspirée des bases de donénes (curseurs)

GÉNÉRALITÉS

- 10 années de travaux internes, collaboration Microsoft Research et composantes appliquées (logiciels)
- Pensé dès le départ pour le machine learning
- Représentation des données (*DataView*) fonctionnelle : immutable, à la demande, inspirée des bases de donénes (curseurs)

ML.NET VS < X >

Fonctionnalités	ML.NET	Scikit-Learn	R/caret	Spark/MLlib
Déploiement	Bon	Passable	mauvais	Impossible
Passage à l'échelle	Très bon	Mauvais	Mauvais	Excellent
Performances	Bon	Passable	Mauvais	Excellent
Mise en route	Bon	Excellent	Excellent	Mauvais
État de l'art	Bon	Bon	Excellent	Médiocre

SCHÉMA GÉNÉRAL

CONSTRUCTION

1. DataView
2. Transfomer/Estimator
3. Fit/Train

UTILISATION DU MODÈLE

2.1 DÉMO 1 : ANALYSE DES SENTIMENTS

[4]

Jeu de données de commentaires/discussion sur wikipedia

INITIALISATION

INITIALISATION

In [8]:

```
open System.Threading
open Microsoft.ML
open Microsoft.ML.AutoML
open Microsoft.ML.Data
open MLCommon
open XPlot.Plotly
open FSharp.Control

//Create the MLContext
let ctx = MLContext()
```

DATAVIEW

TYPAGE ENTRÉES/SORTIES

TYPAGE ENTRÉES/SORTIES

In [9]:

```
/// Type representing the text
/// to run sentiment analysis
[<CLIMutable>]
type SentimentIssue =
    {
        [<LoadColumn(0)>]
        Label : bool

        [<LoadColumn(2)>]
        Text : string
    }
```

TYPAGE ENTRÉES/SORTIES

In [9]:

```
/// Type representing the text
/// to run sentiment analysis
[<CLIMutable>]
type SentimentIssue =
    {
        [<LoadColumn(0)>]
        Label : bool

        [<LoadColumn(2)>]
        Text : string
    }
```

In [10]:

```
[<CLIMutable>]
type  SentimentPrediction =
    {
        [<ColumnName("Predicte
        Prediction : bool;

        Probability : float32;
        Score : float32
    }
```

In [11]:

```
let dataView =
    ctx.Data.LoadFromTextFile<SentimentIssue>(
        @"./Data/sentiment.tsv",
        hasHeader = true)
```

In [12]:

```
let trainTestSplit = ctx.Data.TrainTestSplit(dataView, testFraction=0.2)
let trainingDataView = trainTestSplit.TrainSet
let testDataView = trainTestSplit.TestSet
```

TRANSFORM

TRANSFORM

In [13]:

```
let dataPipeline = ctx.Transforms.Text.FeaturizeText("Features", "Text")
```

In [14]:

```
ConsoleHelper.peekDataViewInConsole<SentimentIssue> ctx trainingDataView dataPi
```

Out[14]:

Label	Text	SamplingKeyColumn	Features
False	" ==He is a Rapist!!!!!== Please edit the article to include this important fact. Thank You. — Preceding unsigned comment added by • "	0.5956414	Sparse vector of size 38828, 152 explicit values
False	The other two films Hitch and Magnolia are also directly related to the community in question, and may be of interest to those who see those films. So why not link to them?	0.5883768	Sparse vector of size 38828, 172 explicit values

In [15]:

```
ConsoleHelper.peekVectorColumnDataInConsole ctx "Features" trainingDataView dat
```

Out[15]:

```
["0.077615050.077615050.077615050.077615050.077615050.077615050.15523010.
077615050.077615050.15523010.077615050.077615050.077615050.077615050.0776
15050.077615050.077615050.077615050.077615050.077615050.23284520.07761505
0.077615050.077615050.077615050.077615050.077615050.077615050.077615050.0
77615050.077615050.077615050.077615050.15523010.077615050.077615050.07761
5050.23284520.077615050.077615050.077615050.077615050.077615050.07761505
0.077615050.077615050.077615050.15523010.077615050.077615050.077615050.07
7615050.077615050.077615050.077615050.077615050.077615050.077615050.07761
5050.077615050.077615050.077615050.077615050.077615050.077615050.07761505
0.077615050.077615050.077615050.15523010.077615050.077615050.077615050.07
7615050.077615050.077615050.077615050.077615050.077615050.077615050.07761
5050.077615050.077615050.077615050.077615050.077615050.077615050.07761505
0.077615050.077615050.077615050.077615050.077615050.077615050.077615050.0
77615050.077615050.077615050.077615050.077615050.077615050.077615050.0776
15050.077615050.077615050.077615050.15523010.077615050.077615050.07761505
0.077615050.077615050.077615050.077615050.077615050.077615050.077615050.0
77615050.077615050.077615050.077615050.077615050.077615050.15523010.07761
5050.077615050.077615050.077615050.0776150500000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000
```

ESTIMATOR

ESTIMATOR

In [16]:

```
let trainer = ctx.BinaryClassification.Trainers.FastTree(labelColumnName = "Lab
                                                         featureColumnName = "F
let learningPipeline = dataPipeline.Append(trainer)
```

FIT

FIT

In [17]:

```
let model = learningPipeline.Fit(trainingDataView)
```

ÉVALUATION

ÉVALUATION

In [18]:

```
let predictions = model.Transform testDataView
let metrics = ctx.BinaryClassification.Evaluate(predictions, "Label", "Score")
ConsoleHelper.printBinaryClassificationMetrics (trainer.ToString()) metrics
```

```
*************************************************************
*       Metrics for Microsoft.ML.Trainers.FastTree.FastTreeBinaryTrainer
binary classification model
*------------------------------------------------------------
*       Accuracy: 94.31%
*       Area Under Curve:      92.00%
*       Area under Precision recall Curve:    77.29%
*       F1Score:  66.67%
*       LogLogg:  0.43%
*       LogLossreduction:  0.14%
*       PositivePrecision:      0.94
*       PositiveRecall:      0.52
*       NegativePrecision:      0.94
*       NegativeRecall:      1.00
*************************************************************
```

PRÉDICTION

PRÉDICTION

In [19]:

```
let predictionEngine = ctx.Model.CreatePredictionEngine<SentimentIssue, Sentime
let sampleStatement = { Label = false; Text = "your moma is fat" }
predictionEngine.Predict(sampleStatement)
```

Out[19]:

```
{Prediction = true;
 Probability = 0.967896342f;
 Score = 8.51538944f;}
```

2.2 DÉMO 2 : AUTOML SUR DÉTECTION DE FRAUDE À LA CARTE BANCAIRE

In [20]:

```
//data from here: https://www.kaggle.com/mlg-ulb/creditcardfraud
let trainFile = @"./Data/creditcard.csv"
let labelCol = "Class"

let colSelection = ctx.Auto().InferColumns(trainFile, labelCol)
let textLoader = ctx.Data.CreateTextLoader(colSelection.TextLoaderOptions)
let trainView = textLoader.Load(trainFile)
```

In [21]:

```
let cts = new CancellationTokenSource()

let expSettings = BinaryExperimentSettings()
expSettings.OptimizingMetric <- BinaryClassificationMetric.AreaUnderRocCurve
expSettings.CancellationToken <- cts.Token

let experiment = ctx.Auto().CreateBinaryClassificationExperiment(expSettings)
```

In [22]:

```
// Ad hoc type for our plot report
type TrainerAuc = { Model: string; AreaUnderRocCurve: float; Runtime: float }

// Progress report handler
let progressReportAsyncSeq = AsyncSeqSrc.create<TrainerAuc>()
let progressReport =
    {new System.IProgress<RunDetail<Data.BinaryClassificationMetrics>> with
        member x.Report v =
            let auc = v.ValidationMetrics.AreaUnderRocCurve
            AsyncSeqSrc.put { Model = v.TrainerName; AreaUnderRocCurve = auc; R
    }
```

In [23]:

```
// We want to plot a AUC/Runtime 2D plot for our models
let plotScatterModel (l : TrainerAuc list) =
    let toPlot =
        l
        |> Seq.map (fun x -> (x.Runtime,x.AreaUnderRocCurve))
    Scatter(
        x = (toPlot |> Seq.map fst),
        y = (toPlot |> Seq.map snd),
        mode= "markers+text",
        textposition = "bottom center",
        text = (l |> Seq.map (fun x -> x.Model))
    )
    |> Chart.Plot
    |> Chart.WithLayout (Layout(
                                title = "Best Models for Credit Card Fraud DataS
                                xaxis = Xaxis(title= "Runtime in seconds"),
                                yaxis = Yaxis(title= "Area under ROC Curve")))
```

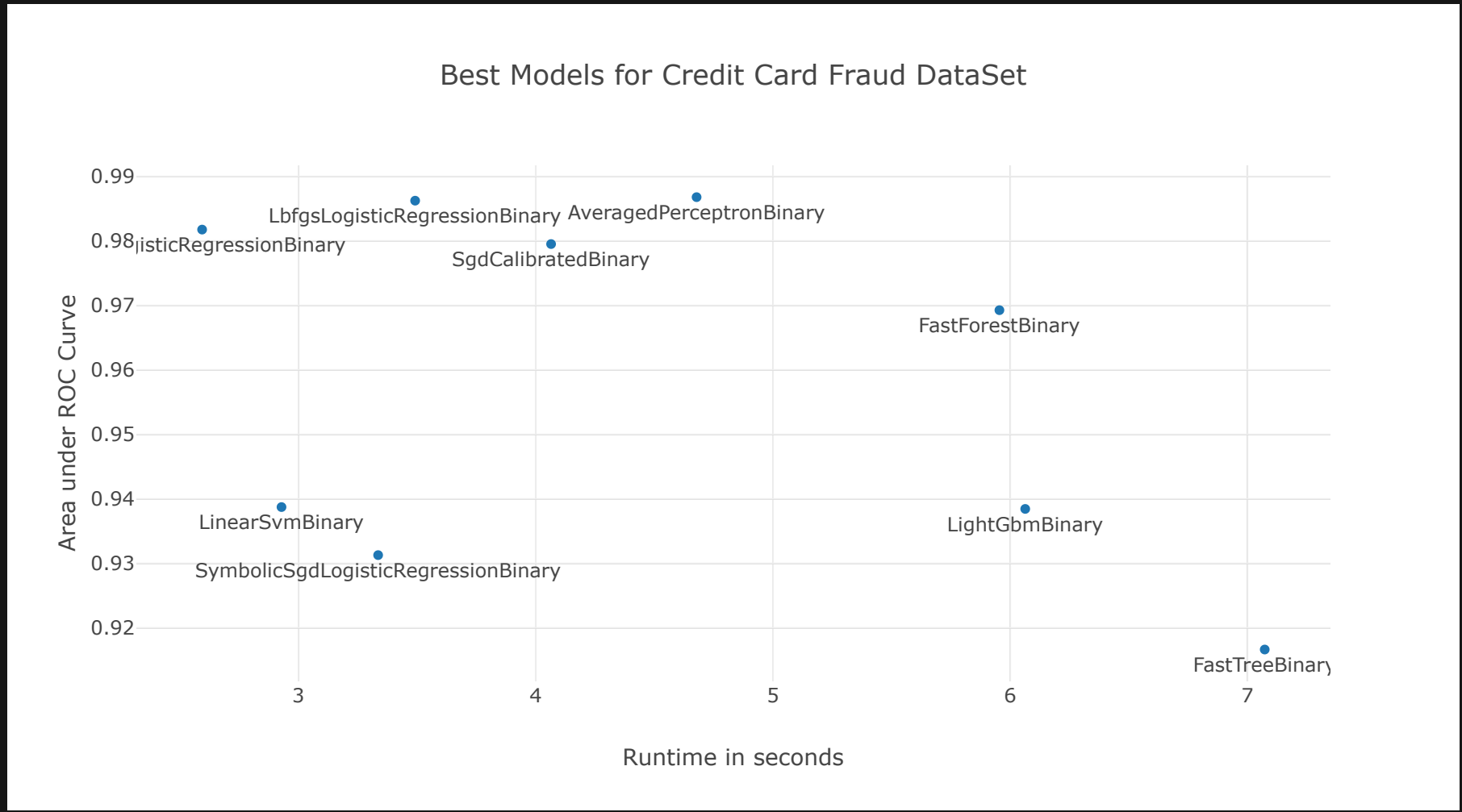
In [24]:

```
let computeAndDraw() =
    async {
        // This starts the automl experiment
        let r = experiment.Execute(trainView,
                                   colSelection.ColumnInformation,
                                   progressHandler=progressReport)
        // Closes the output sequence
        AsyncSeqSrc.close progressReportAsyncSeq
        // Saves the best model
        ctx.Model.Save(r.BestRun.Model, trainView.Schema, "./MLModels/CreditCar
        } |> Async.Start
    // Plots the results in realtime
    AsyncSeqSrc.toAsyncSeq progressReportAsyncSeq
    |> AsyncSeq.scan (fun res auc -> auc::res) []
    |> AsyncSeq.map plotScatterModel
```

In [25]:

```
computeAndDraw()
```

Out[25]:



In [26]:

```
cts.Cancel()    // cancel to stop the experiment and saves the current best mod
```

REFERENCES

[1] Syme Don, ``_The Early History of F# (HOPL IV-second draft)_'', , vol. , number , pp. , 2019. online

[2] Syme Don, Battocchi Keith, Takeda Kenji *et al.*, ``_Strongly-typed language support for internet-scale information sources_'', Technical Report MSR-TR-2012--101, Microsoft Research, vol. , number , pp. , 2012.

[3] Interlandi Matteo, Matusevych Sergiy, Amizadeh Saeed *et al.*, ``_Machine Learning at Microsoft with ML. NET_'', , vol. , number , pp. , 2018.

[4] E. Wulczyn, N. Thain and L. Dixon, ``_Ex machina: Personal attacks seen at scale_'', Proceedings of the 26th International Conference on World Wide Web, 2017.