Ateliers

Formation Machine Learning d’Elastic Stack

Pré-requis :
Poste développeur avec accès réseau Internet libre
Linux (Recommandé) ou Windows 10
Minimum 16 Go RAM
Pré-installation de :

- Git

Atelier 1 : Installation, activation de licence d’évaluation

1.1 Installation et démarrage

Récupérer les distributions fournies d’ElasticSearch et de Kibana, les dézipper.

Démarrer ElasticSearch :
$ES_HOME/bin/elasticsearch

Démarrer Kibana
$KIBANA_HOME/bin/kibana

1.2 Activation License

Connectez-vous à *localhost:5601* et *Management → License management*

Activer la licence d’évaluation

Atelier 2 : Tutoriel

2.1 Chargement des données

Les données représente les réponses d'un service. Les données sont déjà agrégées. On a :

- *timestamp*
- *accept* : Le nombre de requêtes acceptées
- *deny* : le nombre de requêtes refusées
- *total* : le nombre total de requêtes
- *response* : le temps de réponse moyen
- *host* : le hôte
- *service* : le nom du service

Récupérer l'archive fournie et la décompresser :

```
tar -zxvf server_metrics.tar.gz
```

Mettre en place le mapping adéquat :

```
PUT server-metrics
{
  "settings": {
    "number_of_shards":1,
    "number_of_replicas":0
  },
  "mappings": {
      "properties":{
        "@timestamp": {
          "type":"date"
        },
        "accept": {
          "type":"long"
        },
        "deny": {
          "type":"long"
        },
        "host": {
          "type":"keyword"
        },
        "response": {
          "type":"float"
        },
        "service": {
          "type":"keyword"
        },
        "total": {
          "type":"long"
        }
      }
    }
}
```

Après l'avoir rendu exécutable, Utiliser le script fourni pour importer les données.
./upload_server_metrics_noauth.sh

Définir ensuite dans Kibana un index pattern ***server-metrics****

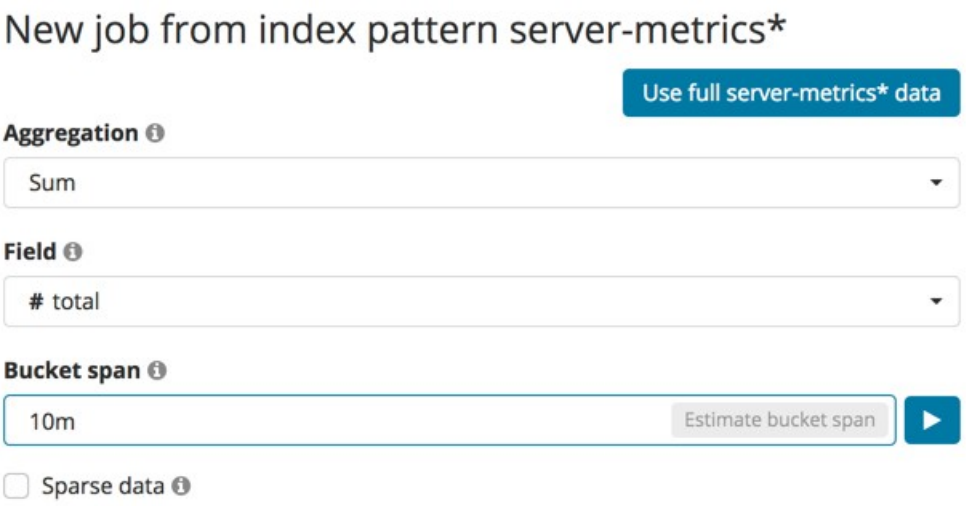
Utiliser le *Data Visualizer* pour visualiser les données.

Sur quelle période porte les données ?

2.2 Création d'un Single Metric Job

Dans Kibana :
Machine Learning → Create new job

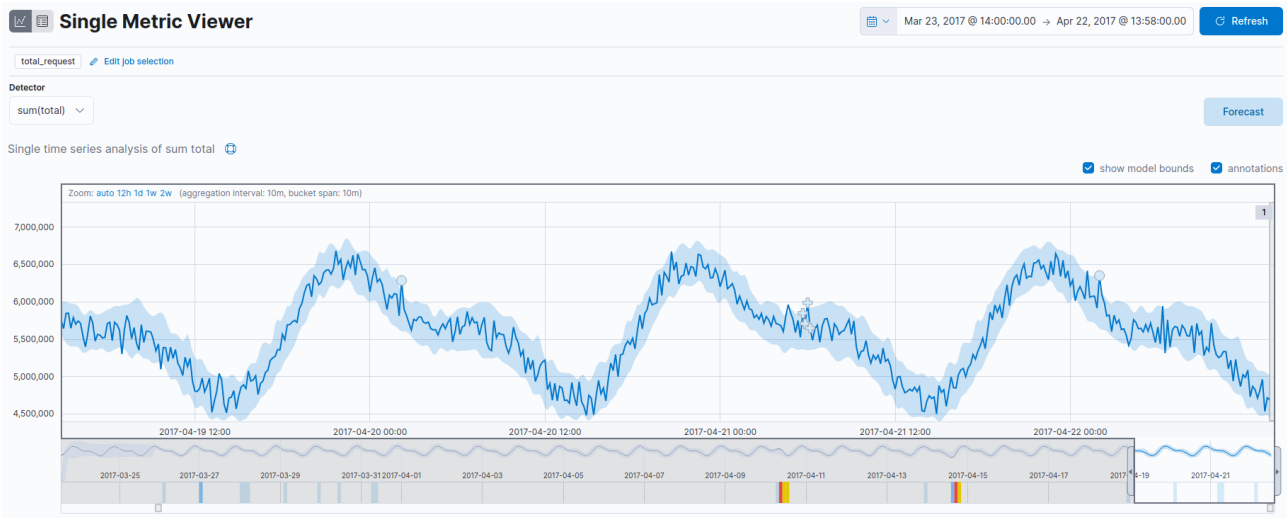
Sélectionner l'index pattern et *Single Metric Job* puis :



Utilisez toutes les données disponibles pour l'analyse
Donner un nom au job, par exemple *total_request* , un groupe par exemple *formation*

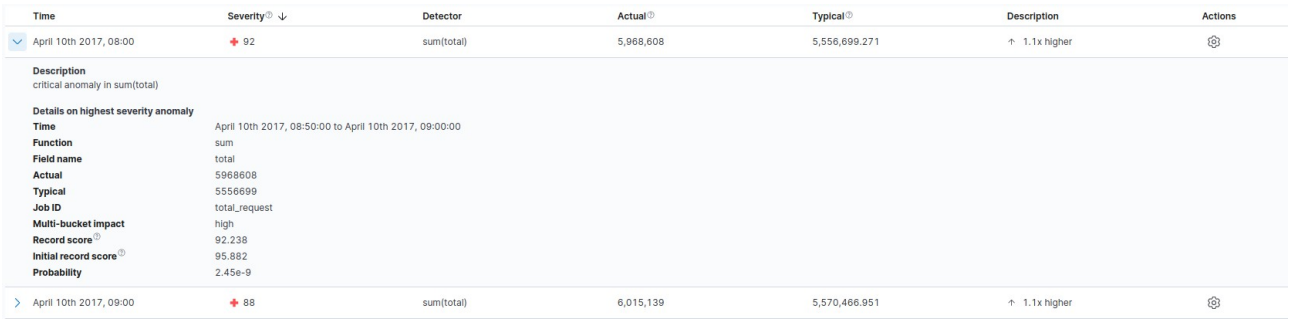
2.3 Visualisation des résultats

Visualiser les résultats dans le ***Single Metric Viewer***



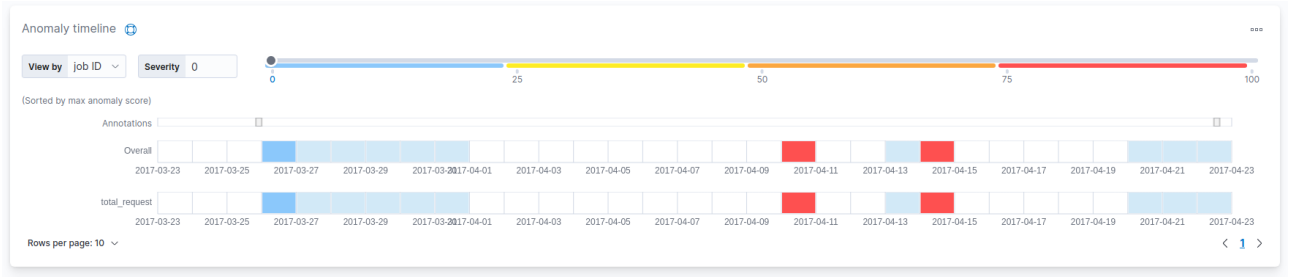
Faire glisser le sélecteur de temps pour sélectionner une section contenant une anomalie critique.

Visualiser les informations liées à l’anomalie, dans le tableau en bas



Visualiser ensuite les résultats avec l’ ***Anomaly Explorer***

Sélectionner une anomalie critique



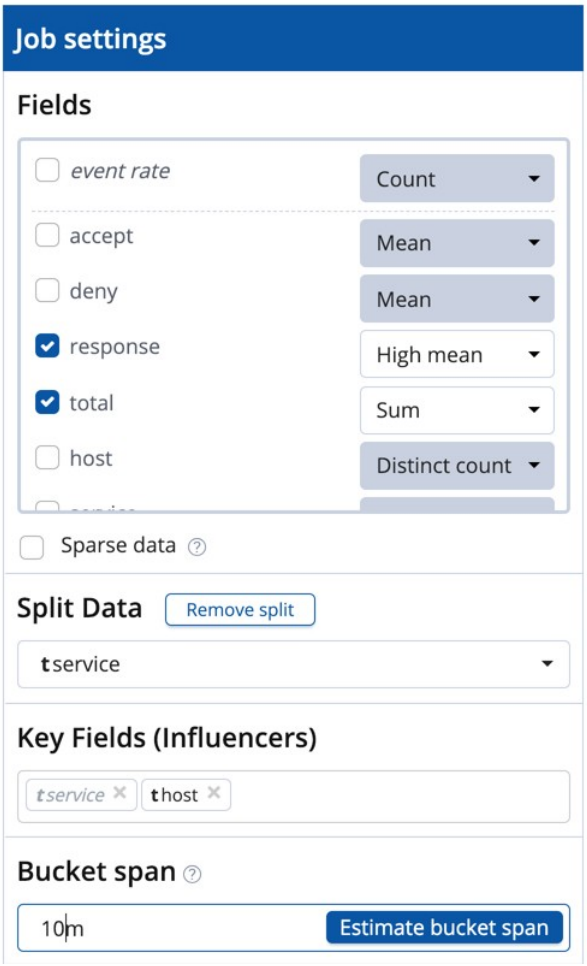
2.4 Ajout d’un job multi-métriques

Démarrer l’assistant pour créer un job multi-métriques et le configurer comme suit :

La configuration consiste en :

- 2 détecteurs
- 1 bucket span de 10 minutes
- Une catégorisation par le champ service
- Un influenceur supplémentaire host

Lui donner le nom de *response_request_by_service*, en utilisant le groupe *formation*



2.5 Visualisation des résultats

Visualiser les résultats dans l'explorateur d'anomalies

Identifier les services et hosts les plus responsables des anomalies

Changer la fenêtre temporelle pour zoomer sur les anomalies

2.6 Prévisions

Reprendre le premier job *total_request* et effectuer une prévision d'1 semaine

Atelier 3. Détection de changement

3.1 Importer les données et visualisez avec Kibana

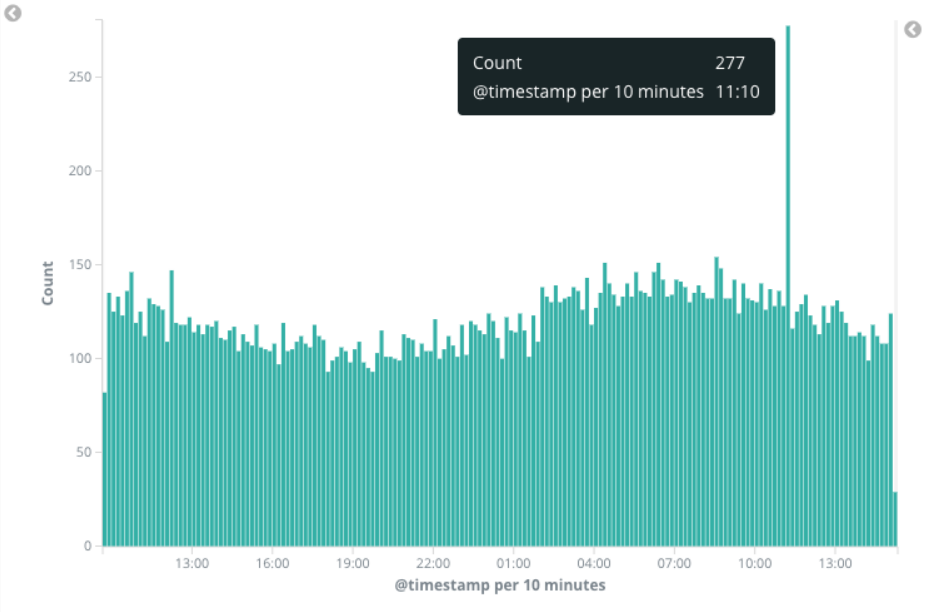
Importer les données avec :
```
curl -X POST -H "Content-Type: application/json"
http://localhost:9200/farequote/_bulk --data-binary "@farequote.json"
```

Créer un index pattern *farequote*

Créer une visualisation Kibana de type vertical Bar avec comme configuration :

- Date histogram sur la minute

Vous devez obtenir quelque chose comme suit :



3.2 Jobs Single Metric et bucket span

Créer un job single metric ***event_high_count_10*** qui détecte l'anomalie en utilisant la fonction *high_count* et un niveau d'agrégation de 10m

Créer un second job ***event_high_count_60*** à l'identique en changeant juste le *bucket span* à 60m

3.3 Catégorisation du comptage

Créer un job multi-metric ***event_high_count_by_airplane*** qui utilise la même fonction de comptage et qui catégorise les données via la compagnie aérienne.
Identifier la compagnie aérienne responsable de l'anomalie

3.4 Ingestion de données avec logstash

Installer logstash avec la distribution fournie et regarder les fichiers de configuration permettant de démarrer 2 pipelines qui traite des fichiers de logs :

- Logs d'accès d'un site web
- Logs applicatifs d'une application métier

Pour démarrer correctement *logstash*, vous devez :

- Placer le fichier *pipelines.yml* dans le répertoire config de *logstash* et l'adapter à votre environnement (chemins des autres fichiers de configuration)
- Éditer les 2 fichiers de configuration *web/config_apache.conf* et *appli/logstash-grok-spring-boot.conf* et modifier les chemins indiqués

Ensuite :

```
${LOGSTASH_HOME}/bin/logstash -r
```

3.5 Analyse de population

A partir des logs d'accès d'un site web, nous recherchons des adresses IP agissant comme des robots automatisés

La configuration du job compare le nombre de requêtes Web par unité de temps, splitté par statut HTTP, par rapport à une population d'IP client

3.6 Rareté

Sur les mêmes données, tentez de trouver les adresses IP qui fréquemment demande des URLs rares

3.7 Catégorisation des messages de traces

Utiliser les données de l'application métier pour détecter des anomalies de décompte dans les messages de logs

4. API et détécteurs

4.1 Comparaison *by_field_name* et *partition_name*

Par l'API, créer 2 jobs qui partitionnent les données d'accès web, pour un bucket de 1j par pays :

- Soit en utilisant *by_field_name*
- Soit en utilisant *partition_name*

Les résultats sont stockés dans un index particuliers

Des influenceurs sur *verb,request*

Utiliser l'API results

4.2 Création de job multi-détecteurs

Via l'API, créer un job :

- bucket span de 10 minutes
- Détecteurs
  - Pays rare
  - Anomalie dans la géo-localisation
- Influenceur : Code statut et Verbe HTTP

Créer le datafeed permettant de transformer les coordonnées de géo-localisation en limitant aux requêtes GET et POST

exécuter le Job

4.3 Règle personnalisée

Utiliser une règle personnalisée pour filtrer les pays comme Monaco, Seychelles, Panama, Luxembourg

5. Analyse de cause et corrélation des données des jobs

Les données disponibles sont :

- Le volume de transactions (KPI) sommé par minutes
- Logs applicatifs (messages textuels semi-structurés) du moteur de traitement des transactions
- Mesures de performance d'utilisation du réseau

5.1 Importer les données

KPI :
Créer l'index avec le bon mapping

```
PUT /it_ops_kpi
{
  "settings" : {
    "number_of_shards" : 1,
    "number_of_replicas" : 1
  },
  "mappings": {
     "doc" : {
      "properties" : {
        "@timestamp" : {
          "type" :  "date",
          "format": "epoch_millis"
        }
} } } }
```

Puis importer les données:

```
curl -X POST -H "Content-Type: application/json"
http://localhost:9200/it_ops_kpi/_bulk --data-binary "@it_ops_kpi.json"
```

Traces du moteur

```
PUT /it_ops_app
{
  "settings" : {
    "number_of_shards" : 1,
    "number_of_replicas" : 1
  },
  "mappings": {
     "doc" : {
      "properties" : {
        "@timestamp" : {
          "type" :  "date",
          "format": "epoch_millis"
        }
```

```
} } } }
```

Importer les données avec :
```
curl -X POST -H "Content-Type: application/json"
http://localhost:9200/it_ops_app/_bulk --data-binary "@it_ops_app_logs.json"
```

Réseau

Créer l'index *pour les données réseau :*
```
PUT /it_ops_network
{
  "settings" : {
    "number_of_shards" : 1,
    "number_of_replicas" : 1
  },
  "mappings": {
     "doc" : {
      "properties" : {
        "@timestamp" : {
          "type" :  "date",
          "format": "epoch_millis"
        }
} } } }
```

Puis :
```
curl -X POST -H "Content-Type: application/json"
http://localhost:9200/it_ops_network/_bulk --data-binary "@it_ops_network.json"
```

5.2 Jobs ML

Créer 3 jobs :

- Détection d'anomalie sur des nombre de transactions faibles
- Anomalie sur le décompte des messages de traces catégorisés
- Anomalies sur les performances réseau (moyennes sur les métriques)

Utiliser un bucket span de 15m et des influenceurs partagés (*hostname*, *physical host*)

5.3 Corrélation des analyses

Identifier les influenceurs responsable de l'anomalie

6. Alertes, Prévisions

6.1 Flux de données continu

Activer la fonction de monitoring de votre cluster.

Créer un index pattern ***.monitoring-es-****

6.2 Mise en place par l'UI

Créer un job multi-métrique surveillant les métriques relatifs à l'index, séparé par index.

6.3 Prévisions

Essayer de prévoir via l'api de forecast quand le nombre de document de l'index ***.monitoring-es**** dépassera une certaine valeur

7. Détection de valeurs anormales

https://www.elastic.co/guide/en/machine-learning/current/ecommerce-outliers.html

Ajouter les données exemple ***eCommerce***

Créer une transformation :

```
PUT _data_frame/transforms/ecommerce-customer-sales
{
  "source": {
    "index": [
      "kibana_sample_data_ecommerce"
    ]
  },
  "pivot": {
    "group_by": {
      "customer_full_name.keyword": {
        "terms": {
          "field": "customer_full_name.keyword"
        }
      }
    },
    "aggregations": {
      "products.quantity.sum": {
        "sum": {
          "field": "products.quantity"
        }
      },
      "products.taxful_price.sum": {
        "sum": {
          "field": "products.taxful_price"
        }
      },
      "order_id.value_count": {
        "value_count": {
          "field": "order_id"
        }
      }
    }
  },
  "description": "E-commerce sales by customer",
  "dest": {
    "index": "ecommerce-customer-sales"
  }
}
```

La démarrer :

```
POST _data_frame/transforms/ecommerce-customer-sales/_start
```

Vous pouvez voir le résultat dans ***Stack Management → Transform***

Créer un job d'analyse de type « Outlier detection»

Par l'assistant ou via l'API :

```
PUT _ml/data_frame/analytics/ecommerce
{
  "source": {
    "index": "ecommerce-customer-sales"
  },
  "dest": {
    "index": "ecommerce-outliers"
  },
  "analysis": {
    "outlier_detection": {
    }
  },
  "analyzed_fields" : {
    "includes" :
["products.quantity.sum","products.taxful_price.sum","order_id.value_count"]
  }
}
POST _ml/data_frame/analytics/ecommerce/_start
```

Le score ***ml.outlier*** est une valeur comprise entre 0 et 1. Plus la valeur est élevée, plus il est probable qu'il s'agisse d'une valeur aberrante
Chaque document est également annoté avec les valeurs d'influence de caractéristiques pour chaque champ.
Ces valeurs totalisent 1 et indiquent quels champs sont les plus importants pour décider si une entité est une valeur aberrante ou pas
Kibana fournit également une matrice de nuages de points . Les valeurs aberrantes avec un score qui dépasse le seuil sont mises en évidence

8. Régression

https://www.elastic.co/guide/en/machine-learning/current/flightdata-regression.html

Ajouter les données exemple ***Flight Data***

On veut prévoir le champ numérique `FlightDelayMins`

Les données sont déjà prêtes pour une analyse, nous n'avons pas besoin de transformations

Créer une analyse de type régression sur ce jeu de données :

- Améliorez éventuellement la qualité de l'analyse en ajoutant une requête qui supprime les données erronées. Dans ce cas, nous omettons les vols d'une distance de 0 kilomètre ou moins.
- Choisir FlightDelayMin comme variable dépendante,
- Ajoutez Cancelled, FlightDelay et FlightDelayType à la liste des champs exclus. Ces champs seront exclus de l'analyse. Il est recommandé d'exclure les champs qui contiennent des données erronées ou qui décrivent la variable dépendante.

Spécifier une valeur de 5 dans le champ *feature importance*
Un modèle mémoire limité à 50 Mo
Un job ID : `model-flight-delay-regression`

```
API :
PUT _ml/data_frame/analytics/model-flight-delays-regression
{
  "source": {
    "index": [
      "kibana_sample_data_flights"
    ],
    "query": {
      "range": {
        "DistanceKilometers": {
          "gt": 0
        }
      }
    }
  },
  "dest": {
    "index": "model-flight-delays-regression"
  },
  "analysis": {
    "regression": {
      "dependent_variable": "FlightDelayMin",
      "training_percent": 90,
      "num_top_feature_importance_values": 5
    }
  },
  "analyzed_fields": {
    "includes": [],
    "excludes": [
      "Cancelled",
```

```
      "FlightDelay",
      "FlightDelayType"
    ]
  }
}
```

Démarrage du job :

```
POST _ml/data_frame/analytics/model-flight-delays-regression/_start
```

Suivre l'évolution du job et accéder aux résultats lorsqu'il est terminé.

```
GET _ml/data_frame/analytics/model-flight-delays-regression/_stats
```

Visualisation des résultats

Les résultats affichent :

- Le contenu de l'index de destination dans un format tabulaire.
- les métriques d'évaluation du modèle,
- les valeurs des Feature importance
- et une matrice de nuage de points

Le tableau de résultat montre une colonne pour la variable dépendante (*FlightDelayMin*), qui contient les valeurs de vérité terrain et les valeurs de prédiction (***ml.FlightDelayMin_prediction***) et une colonne. Il indique également si le document a été utilisé dans l'ensemble d'apprentissage (***ml.is_training***).

Le panneau *Feature Importance* indique comment chaque champ affecte une prédiction particulière. Seules les 5 valeurs les plus significatives sont stockées dans l'index. Cependant, les métadonnées du modèle entraîné contiennent également l'amplitude moyenne des valeurs d'importance des caractéristiques pour chaque champ sur toutes les données d'entraînement.

Les valeurs des f*eature importance* pour chaque prédiction peuvent être visualisées sous forme de graphique

Kibana affiche également une matrice de nuage dans le résultat

Évaluation des résultats

Kibana fournit des métriques **d'erreur de l'entraînement**, qui représentent les performances du modèle sur l'ensemble de données d'entraînement.
Il fournit également des mesures **d'erreur de généralisation**, qui représentent les performances du modèle sur les données de test.

Une erreur quadratique moyenne (MSE) de zéro signifie que les modèles prédisent la variable dépendante avec une précision parfaite
De même, une valeur R-squared de 1 indique que toute la variance de la variable dépendante peut être expliquée par les variables caractéristiques.

```
API :
POST _ml/data_frame/_evaluate
{
 "index": "model-flight-delays-regression",
  "query": {
      "bool": {
        "filter": [{ "term":  { "ml.is_training": true } }]
      }
```

```
    },
  "evaluation": {
    "regression": {
      "actual_field": "FlightDelayMin",
      "predicted_field": "ml.FlightDelayMin_prediction",
      "metrics": {
        "r_squared": {},
        "mse": {},
        "msle": {},
        "huber": {}
      }
    }
  }
}
```

9. Classification

Sur le même jeu de données, on veut prédire si un vol sera retardé ou pas (valeur booléenne)

Créer une analyse de type Classification.

- Choisir FlightDelay comme variable dépendante, qui est le champ que nous voulons prédire avec l'analyse de classification.
- Ajoutez Cancelled, FlightDelayMin et FlightDelayType à la liste des champs exclus.
- Choisir un pourcentage d'entraînement de 10
- Spécifier une valeur pour le nombre de feature importance
- Une limite mémoire
- Un job id : model-flight-delays-classification

Démarrer le job

En API, cela donne :

```
PUT _ml/data_frame/analytics/model-flight-delays-classification
{
  "source": {
    "index": [
      "kibana_sample_data_flights"
    ]
  },
  "dest": {
    "index": "model-flight-delays-classification",
    "results_field": "ml"
  },
  "analysis": {
    "classification": {
      "dependent_variable": "FlightDelay",
      "training_percent": 10,
      "num_top_feature_importance_values": 10
    }
  },
```

```
  "analyzed_fields": {
    "includes": [],
    "excludes": [
      "Cancelled",
      "FlightDelayMin",
      "FlightDelayType"
    ]
  }
}
POST _ml/data_frame/analytics/model-flight-delays-classification/_start
```

Une fois le job terminé, visualisez les résultats

Évaluation des résultats :

Kibana fournit une matrice de confusion normalisée qui contient le pourcentage d'occurrences où l'analyse a classé correctement les points de données avec leur classe réelle et le pourcentage d'occurrences où elle les a mal classées

Kibana fournit également la courbe ROC. Le graphique compare le taux de vrais positifs (axe des y) au taux de faux positifs (axe des x) pour chaque classe ; (dans ce cas vrai et faux). A partir de ce tracé, la valeur de l'aire sous la courbe (AUC) est calculée. C'est un nombre compris entre 0 et 1. Plus l'AUC est élevée, meilleur est le modèle pour prédire correctement les classes