UTOPIOS
TRANSFORMATION DIGITALE INTELLIGENTE

ML.NET

Programme

1. Introduction à ML.NET :
    - Comprendre les principes de base de ML.NET et ses fonctionnalités.
2. Configuration de l'environnement de développement :
    - Installer et configurer les outils nécessaires pour travailler avec ML.NET, tels que Visual Studio ou Visual Studio Code.
3. Préparation des données :
    - Apprendre à préparer et à nettoyer les données avant de les utiliser pour entraîner des modèles.
4. Création d'un premier modèle :
    - Apprendre à créer un modèle simple avec ML.NET pour résoudre un problème de classification ou de régression.
5. Amélioration des modèles :
    - Apprendre à ajuster les hyperparamètres des modèles pour améliorer leurs performances et à évaluer leur qualité à l'aide de différentes mesures.

Programme

6. Traitement du langage naturel :
    - Découvrir comment utiliser ML.NET pour le traitement du langage naturel.
7. Extraction de données à partir de PDF :
    - Apprendre à extraire des données structurées à partir de fichiers PDF.
8. Scraping avec ML.NET :
    - Apprendre à extraire des données non structurées à partir de pages web en utilisant des techniques de scraping avec ML.NET.
9. Apprentissage par renforcement.

Programme

10. Apprentissage non supervisé
11. Déploiement de modèles :
    - Apprendre à déployer les modèles ML.NET dans des applications pour les utiliser en production.

Introduction ML.NET

ML.NET est un framework de machine learning open-source développé par Microsoft spécifiquement pour les applications .NET. Il permet aux développeurs de créer, entraîner et déployer des modèles de machine learning dans leurs applications .NET sans avoir besoin d'expertise approfondie en science des données ou en machine learning.

1. Langages supportés : ML.NET est compatible avec les langages de programmation .NET, tels que C# ou F#. Il permet aux développeurs de rester dans leur écosystème familier tout en tirant parti des capacités de machine learning.
2. Modèles personnalisés : ML.NET offre la possibilité de construire des modèles personnalisés adaptés aux besoins spécifiques de votre application. Vous pouvez utiliser les algorithmes intégrés pour des tâches courantes telles que la classification, la régression, le clustering, la détection d'anomalies et les recommandations.
3. Modèles pré-entraînés : ML.NET intègre des modèles pré-entraînés pour des tâches courantes, ce qui permet de gagner du temps et de simplifier le processus de développement.

Introduction ML.NET

4. Intégration avec TensorFlow et ONNX : ML.NET offre la possibilité d'intégrer des modèles TensorFlow et ONNX, vous permettant ainsi de tirer parti d'un large éventail de modèles pré-entraînés et de solutions développées par la communauté.
5. AutoML : ML.NET inclut un composant appelé AutoML qui facilite la sélection et l'entraînement du meilleur modèle pour vos données en testant automatiquement plusieurs algorithmes et paramètres.
6. Performance et évolutivité : ML.NET est conçu pour être performant et évolutif, ce qui vous permet de traiter de grandes quantités de données et de déployer des modèles de machine learning dans des applications à grande échelle.

Introduction ML.NET

- Quelques domaines où ML.NET peut être appliqué :

1. Classification binaire : Il s'agit de classer des éléments en deux catégories distinctes, comme identifier si un e-mail est un spam ou non.
2. Classification multi-classes : Elle permet de classer des éléments en plusieurs catégories, comme par exemple classer des images d'animaux en différentes espèces.
3. Régression : Prédire une valeur numérique continue, par exemple estimer le prix d'un logement en fonction de ses caractéristiques.
4. Détection d'anomalies : Identifier des comportements anormaux ou des points de données inhabituels, comme détecter une fraude bancaire ou un dysfonctionnement dans un système.
5. Recommandation : Proposer des éléments pertinents en fonction des préférences des utilisateurs, par exemple des films ou des produits.
6. Clustering : Regrouper des éléments similaires ensemble, comme segmenter les clients en fonction de leurs comportements d'achat.
7. Analyse d'image : Identifier et classer des objets dans des images, effectuer de la reconnaissance faciale ou analyser des scènes.

Préparation des données

Apprendre à préparer et à nettoyer les données est une étape cruciale pour garantir la qualité et l'efficacité de vos modèles d'apprentissage automatique.

1. Comprendre les données: Examinez les données en détail pour comprendre les caractéristiques, les types de données, les valeurs manquantes et les erreurs potentielles.
2. Nettoyage des données:
    - Supprimez les doublons: Identifiez et supprimez les enregistrements en double pour éviter les biais.
    - Traitez les valeurs manquantes: Imputez les valeurs manquantes en utilisant des méthodes telles que la moyenne, la médiane ou le mode, ou supprimez les enregistrements contenant des valeurs manquantes si nécessaire.
    - Corrigez les erreurs et les incohérences: Identifiez et corrigez les erreurs de saisie, les valeurs aberrantes et les incohérences dans les données.

Préparation des données

3. Transformation des données:
    - Normalisation/Standardisation: Appliquez des techniques de normalisation ou de standardisation pour mettre les données à l'échelle et réduire les effets des différences d'échelle entre les caractéristiques.
    - Encodage des variables catégorielles: Convertissez les variables catégorielles en variables numériques à l'aide de techniques telles que l'encodage One-Hot ou l'encodage des étiquettes.
    - Réduction de la dimensionnalité: Utilisez des techniques comme l'analyse en composantes principales (ACP) pour réduire la dimensionnalité des données et améliorer l'efficacité du modèle.
4. Sélection des caractéristiques: Identifiez les caractéristiques les plus pertinentes pour l'apprentissage automatique en utilisant des techniques telles que la sélection univariée, la méthode de filtrage ou l'élimination récursive des caractéristiques.
5. Fractionnement des données: Divisez les données en ensembles d'apprentissage, de validation et de test pour évaluer et affiner les performances du modèle.

Préparation des données (démo)

- Supposons que nous ayons un ensemble de données sur les clients d'une banque, et nous voulons prédire si un client souscrira ou non à un dépôt à terme

1. Comprendre les donées :
    - Les colonnes représentent les caractéristiques des clients, et la dernière colonne "subscribed" est notre variable cible (Label).
    - Nous avons des variables numériques et catégorielles. Les variables catégorielles doivent être transformées en variables numériques pour être utilisées dans ML.NET. Les valeurs manquantes et les erreurs potentielles devraient être identifiées et corrigées.
2. Nettoyage des données
    - Dans cet échantillon, nous avons :
        - Un enregistrement en double (la première et la sixième lignes sont identiques)
        - Des valeurs manquantes dans les colonnes "age" et "job" (les deux dernières lignes)

Préparation des données (démo)

3. Transformation des données:

- Créez un pipeline de prétraitement pour transformer les variables catégorielles et normaliser les variables numériques

Rappel Normalisation et standarisation:

- Normalisation: La normalisation met à l'échelle les attributs de manière à ce qu'ils se situent dans une plage de valeurs spécifique, généralement entre 0 et 1. La formule de normalisation est la suivante:

```
x_normalized = (x - x_min) / (x_max - x_min)
```

- Standardisation: La standardisation met à l'échelle les attributs de manière à ce qu'ils aient une moyenne de 0 et un écart-type de 1. La formule de standardisation est la suivante:

```
x_standardized = (x - x_mean) / x_std
```

Préparation des données (démo)

3. Transformation des données:

- Encodage des variables catégorielles:
  - L'encodage One-Hot est une technique utilisée pour représenter des variables catégorielles sous forme numérique. Les variables catégorielles sont celles qui prennent un nombre limité de catégories ou de classes distinctes.

4. Sélection des caractéristiques

- La sélection des caractéristiques est une étape cruciale dans le processus de préparation des données pour l'apprentissage automatique. Elle permet de sélectionner les caractéristiques les plus pertinentes pour un modèle, d'améliorer les performances du modèle, de réduire le temps d'apprentissage et d'éviter le surapprentissage.

Préparation des données (démo)

- Etape 'Cancatenate':
  - Cette étape permet de fusionner les colonnes encodées avec les colonnes numériques restantes en une seule colonne "Features". Cela crée une représentation unifiée des données, prête à être utilisée par les algorithmes d'apprentissage automatique
  - Les algorithmes d'apprentissage automatique de ML.NET s'attendent à recevoir une seule colonne contenant toutes les caractéristiques à utiliser pour l'entraînement

5. Fractionnement des données:
   - Le fractionnement des données consiste à diviser l'ensemble des données en plusieurs sous-ensembles, généralement en ensembles d'apprentissage, de validation et de test. Chacun de ces sous-ensembles est utilisé à différentes étapes du processus d'apprentissage automatique

Préparation des données (Exercice)

Sujet: Prédiction du prix des logements en utilisant la régression avec ML.NET
Le jeu de données contient 14 colonnes au total, dont 13 sont des caractéristiques indépendantes (ou variables explicatives) et une cible dépendante (ou variable à prédire).
Voici la liste des colonnes :

- CRIM: Taux de criminalité per capita par ville.
- ZN: Proportion de terrains résidentiels zonés pour des lots de plus de 25 000 pieds carrés (environ 2 323 mètres carrés).
- INDUS: Proportion d'acres commerciaux non liés au commerce de détail par ville.
- CHAS: Variable factice Charles River (1 si la zone est limitrophe de la rivière, 0 sinon).
- NOX: Concentration en oxydes nitriques (parties par 10 millions).
- RM: Nombre moyen de pièces par logement.
- AGE: Proportion des unités occupées par leur propriétaire construites avant 1940.
- DIS: Distances pondérées jusqu'à cinq centres d'emploi de Boston.
- RAD: Indice d'accessibilité aux routes radiales.
- TAX: Taux d'imposition foncière à la pleine valeur par tranche de 10 000 dollars.
- PTRATIO: Ratio élèves-enseignant par ville.
- B: 1000 (Bk - 0,63)^2 où Bk est la proportion de personnes d'ascendance africaine par ville.
- LSTAT: Pourcentage de la population à statut inférieur.
- MEDV: Valeur médiane des logements occupés par leur propriétaire en milliers de dollars (c'est la variable cible que nous voulons prédire).

Préparation des données (Exercice)

1. Charger l'ensemble de données "Boston Housing" à partir d'un fichier CSV.
2. Diviser les données en ensembles d'apprentissage et de test.
3. Explorer et analyser les données pour détecter les valeurs manquantes, les valeurs aberrantes et les incohérences.
4. Nettoyer les données en éliminant les valeurs manquantes et en traitant les valeurs aberrantes.
5. Créer un pipeline de prétraitement pour normaliser les caractéristiques numériques et, si nécessaire, encoder les caractéristiques catégorielles.

Création d'un premier modèle simple

1. Régression

- Supposons que nous ayon un ensemble de données contenant des informations sur les maisons, telles que la surface habitable, le nombre de chambres et le prix de vente. Nous pouvons utiliser la régression pour prédire le prix de vente d'une nouvelle maison en fonction de ces caractéristiques.

Création d'un premier modèle simple

1. Classification

- Supposons que nous souhaitons entraîner un modèle pour prédire si un fruit donné est une pomme ou une orange en fonction de sa couleur, de sa forme et de son diamètre.

Création d'un premier modèle simple (TP)

Dans ce TP, vous allez développer un modèle de régression pour prédire le niveau de consommation d'énergie d'un bâtiment en fonction de diverses caractéristiques et conditions météorologiques. Vous utiliserez un ensemble de données contenant des informations sur les bâtiments et leur consommation d'énergie pour entraîner et évaluer votre modèle.
Objectifs:

1. Préparez un ensemble de données contenant les informations suivantes sur les bâtiments et leur consommation d'énergie :
   - Surface totale (en mètres carrés)
   - Type de bâtiment (résidentiel, commercial, industriel, etc.)
   - Année de construction
   - Température extérieure moyenne (en degrés Celsius)
   - Humidité relative moyenne (en pourcentage)
   - Niveau de consommation d'énergie (en kWh)
2. Chargez les données dans ML.NET en définissant les classes appropriées pour les caractéristiques et les étiquettes.
3. Divisez les données en ensembles d'apprentissage et de test.

Création d'un premier modèle simple (TP)

4. Entraînez un modèle de régression sur les données d'apprentissage
5. Évaluez les performances du modèle sur l'ensemble de test à l'aide de métriques de régression appropriées.
6. Utilisez le modèle pour prédire le niveau de consommation d'énergie pour de nouveaux bâtiments en fonction de leurs caractéristiques et des conditions météorologiques.

Amélioration des modèles

Les hyperparamètres

1. Comprendre les hyperparamètres: Les hyperparamètres sont des paramètres du modèle qui doivent être définis avant l'entraînement, comme le taux d'apprentissage, le nombre d'itérations, la taille du batch, etc. Il est important de comprendre l'impact de chaque hyperparamètre sur la performance du modèle.
2. Sélectionner les hyperparamètres à optimiser: Les modèles peuvent avoir de nombreux hyperparamètres et tous ne sont pas égaux. Il est important de sélectionner les hyperparamètres les plus importants à optimiser en fonction du problème que vous essayez de résoudre.
3. Définir un espace de recherche: Pour chaque hyperparamètre sélectionné, vous devez définir une plage de valeurs à tester. Par exemple, vous pouvez définir une plage de valeurs pour le taux d'apprentissage entre 0,001 et 0,1 avec des intervalles de 0,001.
4. Définir une mesure de performance: Vous devez définir une mesure de performance qui vous permettra d'évaluer les performances du modèle pour chaque combinaison d'hyperparamètres. Les mesures courantes incluent la précision, le rappel, la F1-score, l'AUC-ROC, etc.

Amélioration des modèles

Les hyperparamètres

5. Effectuer une recherche d'hyperparamètres: Vous pouvez effectuer une recherche d'hyperparamètres en utilisant différentes techniques telles que la recherche en grille, la recherche aléatoire ou la recherche bayésienne. Ces techniques testent différentes combinaisons d'hyperparamètres dans l'espace de recherche défini et sélectionnent la meilleure combinaison en fonction de la mesure de performance définie.
6. Évaluer les performances du modèle: Une fois que vous avez trouvé la meilleure combinaison d'hyperparamètres, vous devez évaluer les performances du modèle sur des données de validation ou de test. Vous pouvez utiliser différentes mesures pour évaluer les performances du modèle, telles que la précision, le rappel, la F1-score, l'AUC-ROC, etc.
7. Interpréter les résultats: Vous devez interpréter les résultats pour comprendre pourquoi le modèle performe mieux avec une combinaison d'hyperparamètres particulière et comment ces résultats peuvent être utilisés pour améliorer le modèle ou résoudre d'autres problèmes similaires.

Amélioration des modèles

Les hyperparamètres démo

1. Comprendre les hyperparamètres: Pour un modèle de classification binaire dans ML.NET, les hyperparamètres importants comprennent le taux d'apprentissage (learningRate), le nombre d'itérations (numberOfIterations), la taille du batch (batchSize), la profondeur de l'arbre (maximumDepth), etc.
2. Sélectionner les hyperparamètres à optimiser.
3. Définir un espace de recherche: Nous allons définir un espace de recherche pour chaque hyperparamètre à optimiser. Par exemple, nous pouvons définir un espace de recherche pour learningRate entre 0,01 et 0,2, pour numberOfIterations entre 100 et 1000, pour batchSize entre 10 et 100, et pour maximumDepth entre 3 et 10.
4. Définir une mesure de performance.

Amélioration des modèles

Validation croisée

La validation croisée est une technique d'entraînement et d'évaluation de modèle qui fractionne les données en plusieurs partitions sur lesquelles elle entraîne plusieurs algorithmes. Cette technique améliore la robustesse du modèle en réservant des données à partir du processus d'entraînement. Outre améliorer les performances sur les observations invisibles, dans les environnements limités en données, cette technique peut être un outil efficace pour entraîner des modèles avec un jeu de données plus petit.

Amélioration des modèles

AutoML

AutoML (Automated Machine Learning) est une fonctionnalité de ML.NET qui permet d'automatiser le processus de sélection du meilleur modèle pour un problème spécifique en explorant et en testant plusieurs algorithmes et combinaisons de paramètres. L'objectif est de simplifier et de réduire le temps nécessaire pour construire et optimiser des modèles de machine learning.
AutoML de ML.NET inclut Model Builder, qui est une interface graphique, ainsi que l'API ML.NET AutoML, qui permet de réaliser ce processus de manière programmatique.

Amélioration des modèles

AutoML

Lorsque vous utilisez AutoML avec ML.NET, vous suivez généralement ces étapes :

1. Préparez vos données : AutoML nécessite des données pour entraîner et tester les modèles. Vous devez préparer un ensemble de données sous forme d'IDataView.
2. Configurez l'expérience AutoML : Spécifiez la tâche de machine learning (classification, régression, etc.), les métriques d'évaluation, et la durée maximale d'exécution de l'expérience.
3. Exécutez l'expérience AutoML : AutoML entraîne et évalue plusieurs modèles avec différentes combinaisons d'algorithmes et d'hyperparamètres.
4. Sélectionnez le meilleur modèle : AutoML retourne le modèle ayant les meilleures performances en fonction de la métrique d'évaluation spécifiée.
5. Utilisez le modèle sélectionné : Vous pouvez utiliser le modèle sélectionné pour effectuer des prédictions sur de nouvelles données.

Traitement du langage naturel

- Le traitement du langage naturel (NLP) est un sous-domaine de l'intelligence artificielle dont l'objectif principal est d'aider les programmes à comprendre et à traiter les données en langage naturel. Le résultat de ce processus est un programme informatique capable de « comprendre » le langage

1. Modèles pré-entraînés : ML.NET offre des modèles pré-entraînés pour des tâches courantes de NLP, comme l'analyse de sentiment, la détection d'entités nommées et l'extraction de phrases clés.
2. Featurisation du texte : ML.NET fournit des outils pour transformer le texte brut en caractéristiques numériques (features) exploitables par les modèles d'apprentissage automatique. Il s'agit notamment du marquage de parties du discours, de la tokenisation et de la suppression des mots vides.
3. Intégration avec d'autres bibliothèques NLP : ML.NET peut être intégré avec d'autres bibliothèques populaires de NLP telles que spaCy, NLTK et Stanford NLP, offrant une plus grande flexibilité pour résoudre des problèmes de NLP complexes.

Traitement du langage naturel BERT

- BERT est un acronyme pour "Bidirectional Encoder Representations from Transformers". Il s'agit d'un modèle de traitement automatique du langage naturel (NLP) développé par Google en 2018. BERT est basé sur l'architecture des Transformers, qui a été introduite par Vaswani et al. en 2017. - Les Transformers sont des modèles qui utilisent un mécanisme d'attention pour capturer les dépendances à long terme dans les données textuelles.
- Le principal avantage de BERT par rapport aux modèles précédents est sa capacité à traiter le contexte bidirectionnel. Cela signifie que BERT prend en compte les mots qui précèdent et suivent un mot donné dans une phrase pour mieux comprendre la signification de ce mot. Cette approche bidirectionnelle est ce qui distingue BERT des modèles unidirectionnels ou context-free, qui ne tiennent compte que d'un seul sens du contexte.
- BERT a été pré-entraîné sur de vastes ensembles de données textuelles (comme Wikipedia) en utilisant deux tâches d'apprentissage non supervisé :
    1. Modélisation du langage masqué (MLM) : dans cette tâche, certaines parties du texte sont masquées, et BERT doit prédire les mots manquants en se basant sur le contexte environnant.
    2. Prédiction de la prochaine phrase (NSP) : dans cette tâche, BERT reçoit deux phrases et doit déterminer si la deuxième phrase suit logiquement la première.

Traitement du langage naturel BERT

- Une fois pré-entraîné, BERT peut être adapté à diverses tâches spécifiques de traitement du langage naturel, comme la classification de texte, la réponse aux questions, la reconnaissance d'entités nommées, et l'analyse des sentiments, en utilisant un processus appelé "fine-tuning" (ajustement fin). Ce processus consiste à entraîner BERT sur un ensemble de données spécifique à la tâche pour un petit nombre d'époques.
- Il est possible d'utiliser BERT avec ML.NET.
- Pour utiliser BERT avec ML.NET, vous pouvez vous appuyer sur la bibliothèque ONNX (Open Neural Network Exchange), qui permet d'intégrer des modèles de deep learning comme BERT dans des applications .NET.

Traitement du langage naturel BERT ONNX

- ONNX est l'acronyme de Open Neural Network Exchange. C'est un format ouvert et standardisé pour les modèles d'apprentissage profond qui vise à faciliter l'interopérabilité entre les différents frameworks et bibliothèques d'apprentissage profond, tels que TensorFlow, PyTorch, MXNet, Caffe2, et Microsoft Cognitive Toolkit.
- L'idée derrière ONNX est de permettre aux développeurs de créer, entraîner et déployer des modèles d'apprentissage profond en utilisant différents frameworks et outils, sans avoir à se soucier de la conversion manuelle des modèles entre les formats spécifiques à chaque framework.

Traitement du langage naturel BERT ONNX

Les principaux avantages d'ONNX sont les suivants :

1. Interopérabilité : ONNX permet de convertir facilement les modèles entre différents frameworks d'apprentissage profond. Cela simplifie le processus de développement et de déploiement des modèles, car les développeurs peuvent choisir le framework le plus adapté à chaque étape du pipeline d'apprentissage profond.
2. Portabilité : Grâce à son format standardisé, ONNX permet de déployer des modèles d'apprentissage profond sur une grande variété de plateformes et de dispositifs, tels que les serveurs, les ordinateurs de bureau, les appareils mobiles, et les dispositifs embarqués.
3. Optimisation : ONNX prend en charge un ensemble de techniques d'optimisation qui peuvent améliorer les performances des modèles d'apprentissage profond lors de l'inférence. Cela inclut la fusion des couches, la quantification, et la réduction de la précision, entre autres.
4. Évolutivité : ONNX est conçu pour évoluer avec le temps et prendre en charge de nouvelles fonctionnalités et architectures de modèles d'apprentissage profond. Cela garantit que les modèles ONNX restent compatibles avec les dernières avancées de la recherche et les mises à jour des frameworks d'apprentissage profond.

Traitement du langage naturel BERT ONNX et ML.NET

Voici les étapes générales pour utiliser BERT avec ML.NET:

1. Convertissez le modèle BERT au format ONNX : Vous devez d'abord convertir le modèle BERT pré-entraîné (en format PyTorch ou TensorFlow) en un modèle ONNX. Vous pouvez utiliser des outils de conversion, tels que tf2onnx (pour les modèles TensorFlow) ou torch.onnx (pour les modèles PyTorch).
2. Installez les packages nécessaires : Pour utiliser ML.NET avec ONNX, installez les packages NuGet suivants dans votre projet .NET:
    - Microsoft.ML
    - Microsoft.ML.OnnxTransformer
    - Microsoft.ML.OnnxRuntime (pour l'exécution du modèle ONNX)
3. Chargez le modèle ONNX dans ML.NET : Utilisez la classe OnnxModel pour charger le modèle ONNX dans votre application .NET.

Traitement du langage naturel BERT ONNX et ML.NET

3. Créez un pipeline de transformation : Créez un pipeline ML.NET qui inclut les étapes de prétraitement des données (comme la tokenisation et la conversion en tenseurs) et l'utilisation du modèle ONNX avec la classe OnnxTransformer.
4. Entraînez et évaluez le modèle : Si vous effectuez un ajustement fin (fine-tuning) pour une tâche spécifique, vous pouvez entraîner le modèle sur votre ensemble de données et évaluer sa performance à l'aide des métriques appropriées.
5. Utilisez le modèle pour effectuer des prédictions : Après avoir entraîné et évalué le modèle, vous pouvez l'utiliser pour effectuer des prédictions sur de nouvelles données