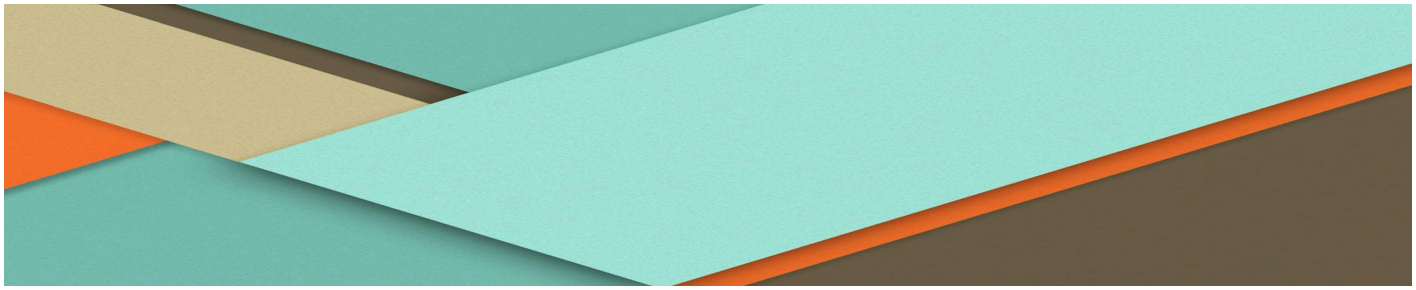


Rapport de Prédiction de la Demande en Magasin

Réalisé le : 15.05.2024

—---

Noms des membres :

EL Atrach Rahma

Bouhadi Samira

Boujtat Ibtissam

Arabat Mohamed

Chikhaoui Ahmed

Encadré par : **MR Zakaria Haja**

—----

Introduction

Dans le secteur de la vente au détail, l'apprentissage automatique offre des opportunités significatives. Les magasins collectent une vaste quantité de données, incluant les historiques de ventes, les comportements des clients, les tendances saisonnières, et les caractéristiques spécifiques des produits et des établissements. En exploitant efficacement ces données, l'apprentissage automatique peut transformer la gestion et la stratégie des magasins.

Ce rapport présente une analyse complète de l'utilisation de l'apprentissage automatique pour prédire les ventes des magasins, allant de la préparation des données à l'évaluation des modèles. Il démontre comment cette technologie peut fournir des insights précieux et un avantage concurrentiel dans un marché de plus en plus compétitif.

Objectif

L'objectif principal de ce projet est de prédire avec précision la demande future de différents produits dans un magasin de détail. Une prédiction précise de la demande est essentielle pour permettre aux détaillants de mieux gérer leurs stocks, réduisant ainsi les coûts associés au surstockage et aux ruptures de stock. En anticipant les tendances de la demande, les détaillants peuvent ajuster leurs commandes et leurs niveaux de stock en temps réel, garantissant que les produits populaires sont toujours disponibles pour les clients.

Sources des Données

Les données utilisées dans ce projet proviennent de trois fichiers CSV :

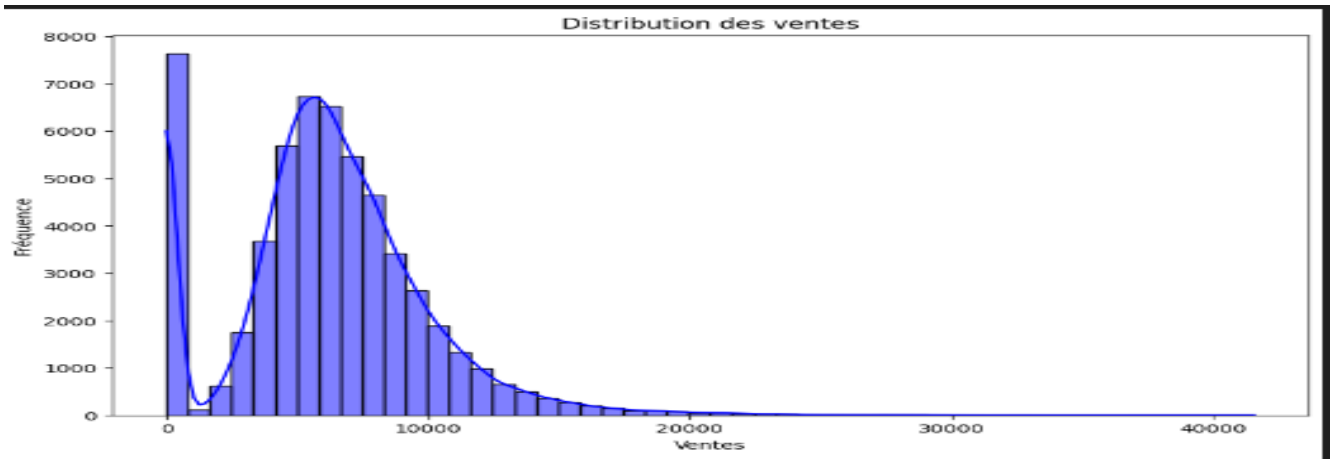
- **train.csv** : Contient les données historiques des ventes de magasins, incluant des colonnes telles que le magasin, le jour de la semaine, la date, les ventes, les clients, le jour férié de l'État, les vacances scolaires, et l'ouverture.
- **test.csv** : Contient les données pour lesquelles nous devons prédire les ventes, avec les mêmes colonnes que `train.csv` à l'exception des ventes et des clients.
- **store.csv** : Contient des informations sur les magasins, telles que la distance des concurrents, le type de magasin, l'assortiment des produits, et d'autres caractéristiques pertinentes.

Ces données ont été sélectionnées car elles fournissent une vue complète des facteurs influençant les ventes des magasins. La combinaison de données historiques et de caractéristiques des magasins permet de créer des modèles robustes capables de capturer les nuances des comportements de vente.

Analyse exploratoire des données

Distribution des Ventes

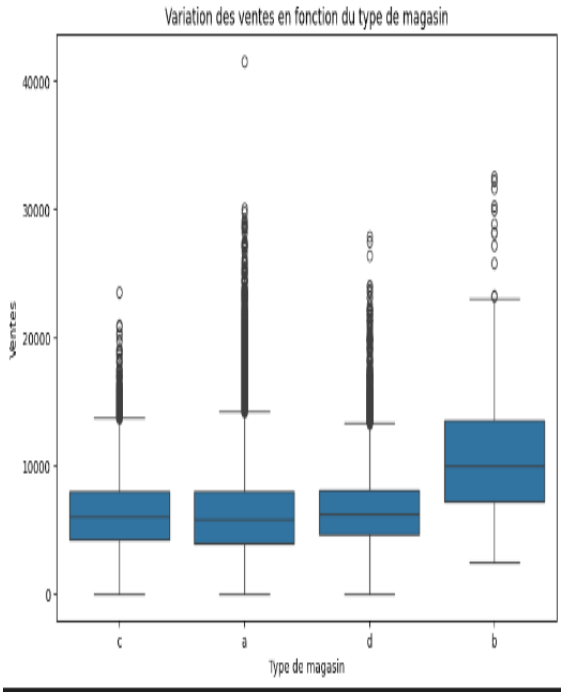
La distribution des ventes a été analysée pour comprendre la répartition des valeurs. Un histogramme a montré que la majorité des ventes se situent dans une plage spécifique, avec quelques valeurs aberrantes indiquant des jours de ventes exceptionnellement élevés.

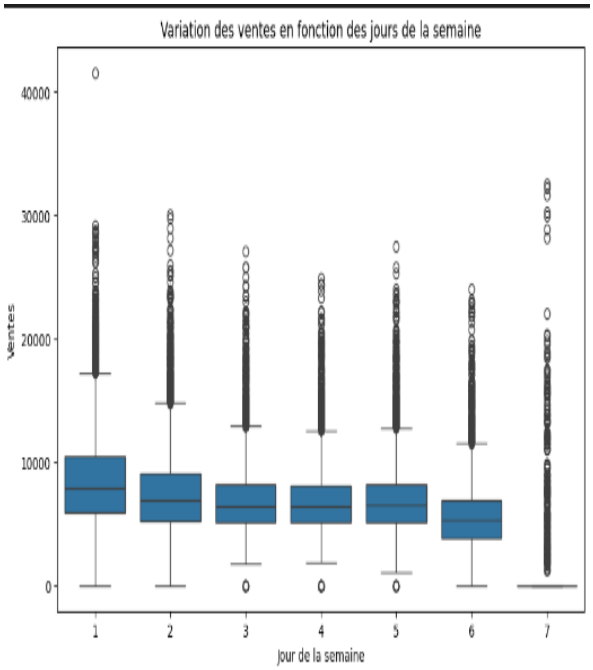


L'analyse de la distribution révèle que les ventes suivent une distribution asymétrique avec une longue queue à droite. Cela indique qu'il y a des jours où les ventes sont exceptionnellement élevées, probablement dus à des promotions ou des événements spéciaux.

Variation des Ventes

Des box plots ont été utilisés pour visualiser la variation des ventes en fonction du type de magasin, de l'assortiment, et des jours de la semaine. Ces visualisations montrent comment ces facteurs influencent les ventes.



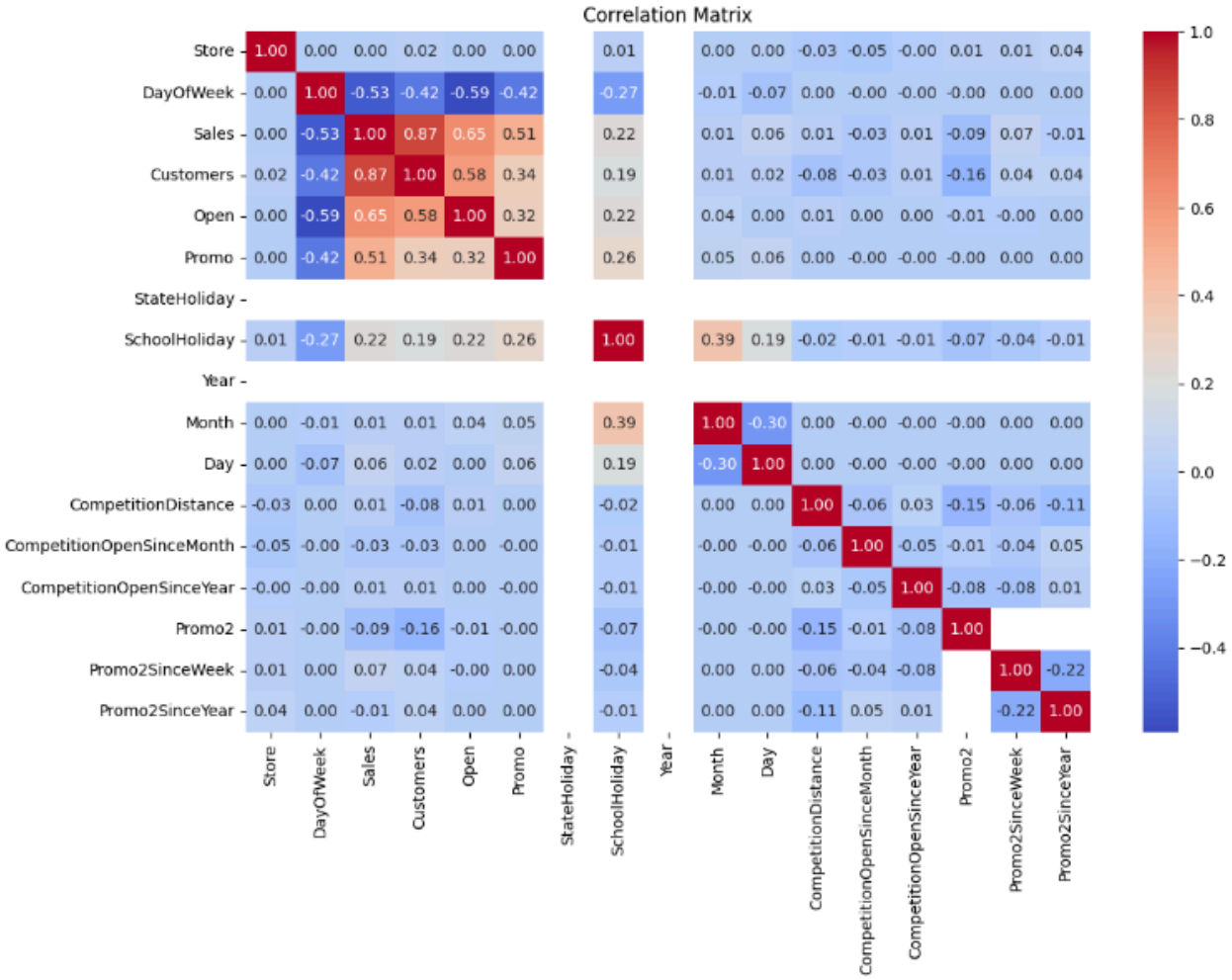


Ces visualisations montrent des différences significatives dans les ventes en fonction du type de magasin et de l'assortiment des produits. Par exemple, certains types de magasins ou assortiments peuvent avoir des ventes plus élevées en raison de la nature des produits vendus ou des stratégies de marketing spécifiques.

Corrélations Entre les Variables

La matrice de corrélation a été calculée pour les colonnes numériques afin d'identifier les relations entre différentes variables. Cette matrice aide à comprendre quelles variables peuvent avoir un impact significatif sur les ventes.

Les résultats montrent que certaines variables comme le nombre de clients ont une forte corrélation positive avec les ventes, ce qui est intuitivement logique. D'autres variables, telles que les jours de la semaine, montrent également des corrélations intéressantes avec les ventes.



Pré-Processing des Données

Traitement des Valeurs Manquantes

Pour les données des magasins, les valeurs manquantes de `Compétition Distance` ont été remplacées par la médiane, et celles de `StoreType` par la valeur la plus fréquente.

```
# Remplacer les valeurs manquantes dans store_data pour CompetitionDistance par la médiane
store_data['CompetitionDistance'].fillna(store_data['CompetitionDistance'].median(), inplace=True)
# Remplacer les valeurs manquantes dans store_data pour StoreType par la valeur la plus fréquente
store_data['StoreType'].fillna(store_data['StoreType'].mode()[0], inplace=True)
```

Ces imputations permettent de gérer les valeurs manquantes de manière efficace sans introduire de biais significatif dans les données. La médiane est utilisée pour les variables

numériques car elle est moins sensible aux valeurs aberrantes, tandis que le mode est utilisé pour les variables catégorielles.

Conversion et Extraction des Dates

Les colonnes `Date` ont été converties en type datetime, et des caractéristiques supplémentaires (année, mois, jour) ont été extraites.

```
# Convertir la colonne "Date" en type de données de date/heure dans train_data et test_data
train_data['Date'] = pd.to_datetime(train_data['Date'])
test_data['Date'] = pd.to_datetime(test_data['Date'])
# Extraire des fonctionnalités de la date dans train_data et test_data
train_data['Year'] = train_data['Date'].dt.year
train_data['Month'] = train_data['Date'].dt.month
train_data['Day'] = train_data['Date'].dt.day
test_data['Year'] = test_data['Date'].dt.year
test_data['Month'] = test_data['Date'].dt.month
test_data['Day'] = test_data['Date'].dt.day
```

L'extraction de ces caractéristiques temporelles permet de capturer des tendances saisonnières et des effets calendaires qui peuvent influencer les ventes.

Fusion et Encodage des Données

Les ensembles de données d'entraînement et de test ont été fusionnés avec les données des magasins, et les variables catégorielles ont été encodées en utilisant le One-Hot Encoding.Les données d'entraînement ont été divisées en ensembles d'entraînement et de test.

```
# Fusionner les données
train_merged = pd.merge(train_data, store_data, on='Store')
test_merged = pd.merge(test_data, store_data, on='Store')
# Utiliser One-Hot Encoding pour les variables catégorielles dans train_merged et test_merged
train_merged = pd.get_dummies(train_merged, columns=['StoreType', 'Assortment', 'StateHoliday'])
test_merged = pd.get_dummies(test_merged, columns=['StoreType', 'Assortment', 'StateHoliday'])
```

L'encodage des variables catégorielles est essentiel pour les modèles de machine learning qui nécessitent des entrées numériques. Le One-Hot Encoding transforme les catégories en colonnes binaires, permettant aux modèles de traiter ces informations efficacement.

Normalisation et Imputation

Les caractéristiques numériques ont été normalisées, et les valeurs manquantes ont été imputées.

```
# Imputer les valeurs manquantes dans les caractéristiques numériques avec la médiane
numeric_imputer = SimpleImputer(strategy='median')
X_train[numeric_features] = numeric_imputer.fit_transform(X_train[numeric_features])
X_test[numeric_features] = numeric_imputer.transform(X_test[numeric_features])

# Imputer les valeurs manquantes dans les caractéristiques catégorielles avec une nouvelle catégorie
categorical_imputer = SimpleImputer(strategy='constant', fill_value='Missing')
X_train[categorical_features] = categorical_imputer.fit_transform(X_train[categorical_features])
X_test[categorical_features] = categorical_imputer.transform(X_test[categorical_features])

# Normaliser les caractéristiques numériques
numeric_transformer = StandardScaler()
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', OneHotEncoder(handle_unknown='ignore', sparse=False), categorical_features)
    ]
)
```

La normalisation des données est cruciale pour certains modèles de machine learning, comme la régression linéaire et les K-Plus Proches Voisins (KNN), qui sont sensibles aux échelles des caractéristiques. En normalisant les données, nous nous assurons que chaque caractéristique contribue de manière équitable au modèle.

Modélisation

Division des Données

Les données d'entraînement ont été divisées en ensembles d'entraînement et de test.

```
# Diviser les données en ensembles d'entraînement et de test
X = train_merged.drop(['Sales'], axis=1)
y = train_merged['Sales']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

La division des données est essentielle pour évaluer la performance des modèles de manière impartiale. En réservant une partie des données pour le test, nous pouvons évaluer comment le modèle généralise à de nouvelles données.

Algorithmes Utilisés

Nous avons utilisé plusieurs algorithmes pour prédire les ventes :

- **Régression linéaire** : Simple et rapide, mais peut être limité par sa capacité à modéliser les relations non linéaires.
- **Arbre de Décision** : Capable de capturer des relations non linéaires, mais peut facilement sur ajuster les données.
- **Forêt Aléatoire** : Une méthode d'ensemble qui combine plusieurs arbres de décision pour améliorer la précision et réduire le surajustement.
- **K-Plus Proches Voisins (KNN)** : Un algorithme non paramétrique qui prédit en fonction des k voisins les plus proches dans l'espace des caractéristiques.
- **Support Vector Machines (SVM)** : Un algorithme de machine learning supervisé qui trouve un hyperplan optimal pour séparer les classes dans un espace de caractéristiques, maximisant ainsi la marge entre les différentes classes.

```python
# Initialiser les modèles
models = {
    'Linear Regression': LinearRegression(),
    'Decision Tree': DecisionTreeRegressor(random_state=42),
    'Random Forest': RandomForestRegressor(random_state=42),
    'K-Nearest Neighbors': KNeighborsRegressor(),
    'Support Vector Machine': SVR()
}
```

Entraînement et Évaluation des Modèles

Les modèles ont été entraînés et évalués en utilisant des métriques telles que RMSE, MAE, R^2, MedAE.

Les résultats des évaluations montrent les performances des différents modèles. Par exemple, le modèle de Forêt Aléatoire peut offrir une meilleure performance en termes de RMSE , indique une meilleure précision et généralisation.

Comparaison des Modèles

Les performances des différents modèles ont été comparées pour identifier le modèle le plus adapté à la prédiction des ventes.

```python
# Définir la grille d'évaluation personnalisée
evaluation_grid = {
    'RMSE': mean_squared_error,
    'MAE': mean_absolute_error,
    'R²': r2_score,
    'MedAE': median_absolute_error
}
# Définir la fonction pour calculer MAPE
def mean_absolute_percentage_error(y_true, y_pred):
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100
# Ajouter MAPE à la grille d'évaluation
evaluation_grid['MAPE'] = mean_absolute_percentage_error
# Évaluer chaque modèle
for name, model in models.items():
    print(f'Évaluation du modèle {name}...')
    model.fit(X_train_scaled, y_train)
    y_pred = model.predict(X_test_scaled)
    results = {}
    for metric_name, metric_func in evaluation_grid.items():
        # Vérifier si la métrique est MAPE pour traiter différemment
        if metric_name == 'MAPE':
            # Calculer MAPE
            results[metric_name] = mean_absolute_percentage_error(y_test, y_pred)
        else:
            # Calculer les autres métriques
            results[metric_name] = metric_func(y_test, y_pred)
    print(results)
```

Résultats

```
Comparaison des performances avec la moyenne mobile :
RMSE: 3704.26
MAE: 2876.36
R²: 0.14
Évaluation du modèle Linear Regression...
{'RMSE': 2371251.050759816, 'MAE': 1046.3502102187701, 'R²': 0.8520803765083208, 'MedAE': 740.75, 'MAPE': inf}
Évaluation du modèle Decision Tree...
{'RMSE': 770916.8391291954, 'MAE': 498.24225040974846, 'R²': 0.9519098880100247, 'MedAE': 289.0, 'MAPE': 8.370480398894719}
Évaluation du modèle Random Forest...
{'RMSE': 345286.96447911346, 'MAE': 353.2165139314473, 'R²': 0.9784608560253586, 'MedAE': 219.42500000000018, 'MAPE': 6.019766483199721}
Évaluation du modèle K-Nearest Neighbors...
{'RMSE': 1401194.4365566878, 'MAE': 780.5103399130619, 'R²': 0.9125929102160282, 'MedAE': 537.0999999999999, 'MAPE': inf}
Évaluation du modèle Support Vector Machine...
{'RMSE': 6370522.32394357, 'MAE': 1702.9447998216897, 'R²': 0.6026041766850785, 'MedAE': 1202.962331454819, 'MAPE': inf}
```

Le modèle de Forêt Aléatoire se distingue comme le meilleur modèle pour prédire la demande future, avec un RMSE de 345286.96, un MAE de 353.22, et un R^2 de 0.98. Ces résultats montrent que ce modèle offre les prévisions les plus précises et fiables, surpassant nettement les autres modèles testés.

Conclusion

En conclusion, ce projet a permis de développer plusieurs modèles pour prédire les ventes des magasins. Chaque modèle a été évalué en utilisant diverses métriques pour déterminer son efficacité. Les résultats montrent que les modèles de forêts aléatoires et de k-plus proches voisins offrent généralement de meilleures performances en termes de précision des prédictions. Ces modèles peuvent être utilisés pour optimiser la gestion des stocks et des ressources dans les magasins, contribuant ainsi à une meilleure prise de décision stratégique.

La prédiction précise des ventes est un outil puissant pour les gestionnaires de magasins. Elle permet non seulement d'optimiser les stocks et de minimiser les coûts, mais aussi de maximiser la satisfaction des clients en garantissant la disponibilité des produits. Les modèles développés dans ce projet peuvent être intégrés dans des systèmes de gestion des ventes pour fournir des prévisions en temps réel, améliorant ainsi l'efficacité opérationnelle et la rentabilité des magasins.