Rapport de Projet : Prédiction des Revenus des Films avec le Machine Learning

1. Définition de l'objectif du projet

L'objectif de ce projet est de développer un modèle de machine learning capable de prédire les revenus des films à partir de différentes caractéristiques disponibles avant la sortie du film. En utilisant des techniques d'apprentissage automatique, nous souhaitons exploiter un dataset comprenant des informations détaillées sur plusieurs films pour entraîner notre modèle à faire des prédictions précises.

2. Source des données

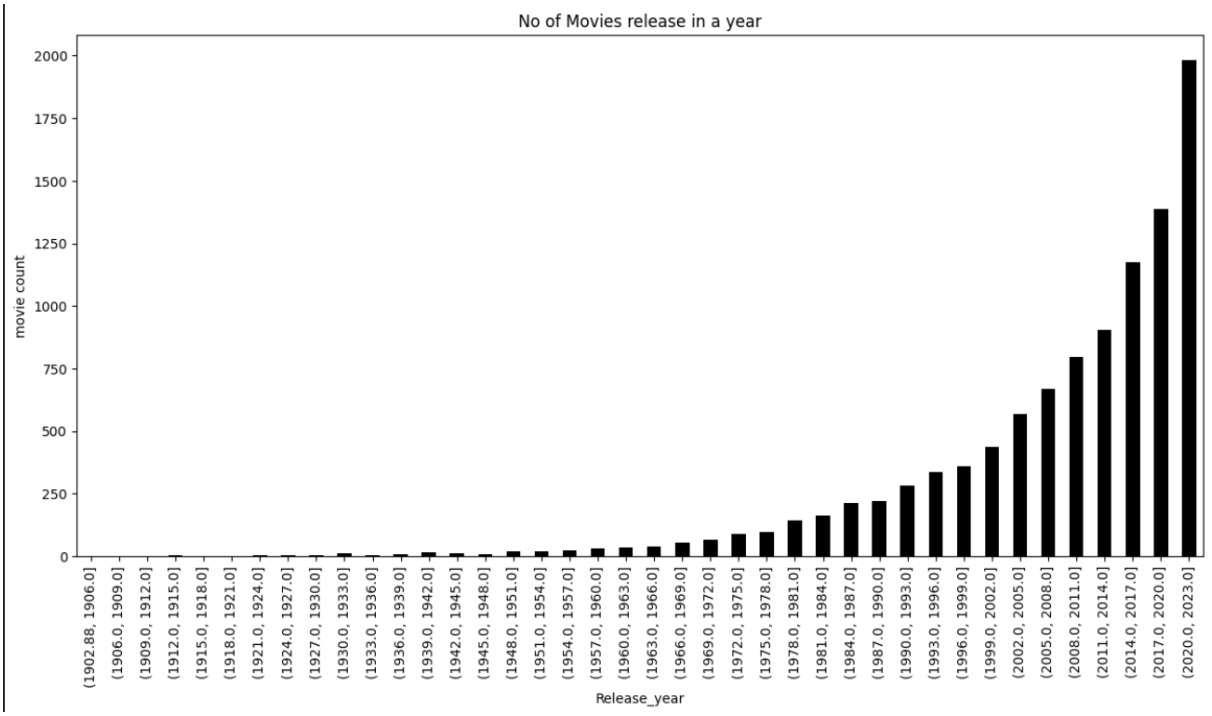
Les données utilisées pour ce projet proviennent d'un dataset disponible sur Kaggle, qui contient des informations complètes sur divers films récents.

	score	budget_x	country	primary_genre	Release_year	Release_month
0	73.0	75000000.0	AU	Drama	2023	3
1	78.0	460000000.0	AU	Science Fiction	2022	12
2	76.0	100000000.0	AU	Animation	2023	4
3	70.0	12300000.0	AU	Animation	2023	1
4	61.0	77000000.0	US	Action	2023	3

3. Analyse exploratoire des données

Cette analyse va nous permettre de comprendre la structure et les caractéristiques des données avant de procéder à la modélisation. Voici les étapes clés de notre EDA :

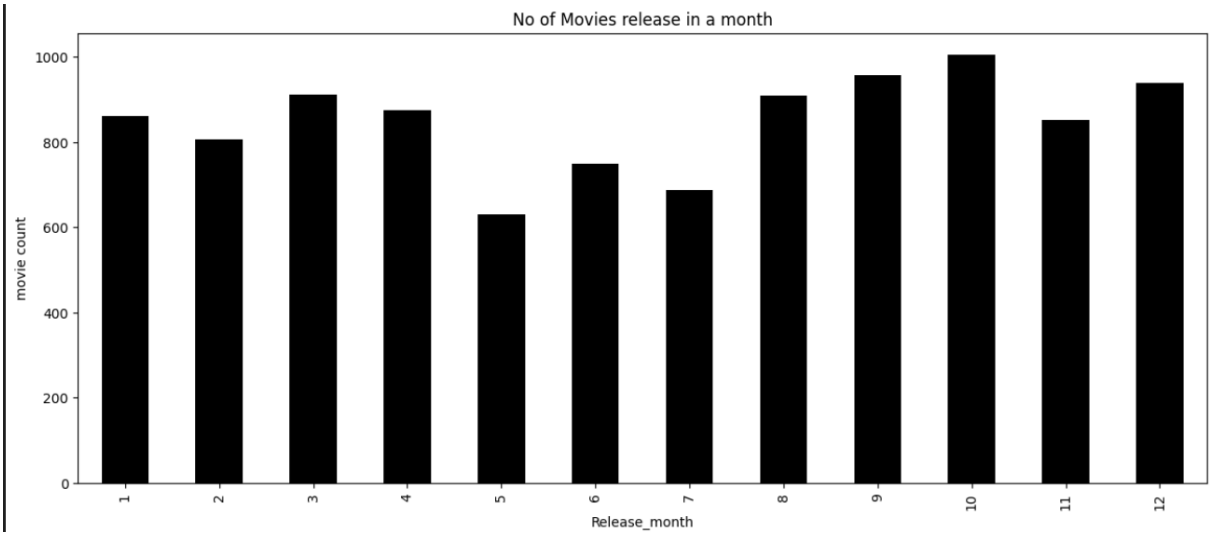
Création d'un diagramme à barres pour visualiser le nombre de films sortis sur différentes périodes:



Cette visualisation nous permet de :

- Identifier des tendances : Voir comment la production de films a évolué au fil du temps.
- Détecter des motifs : Reconnaître les motifs cycliques et les changements significatifs dans l'industrie cinématographique au fil des années.

Création d'un diagramme à barres qui permet de visualiser la répartition des sorties de films par mois au cours de l'année:

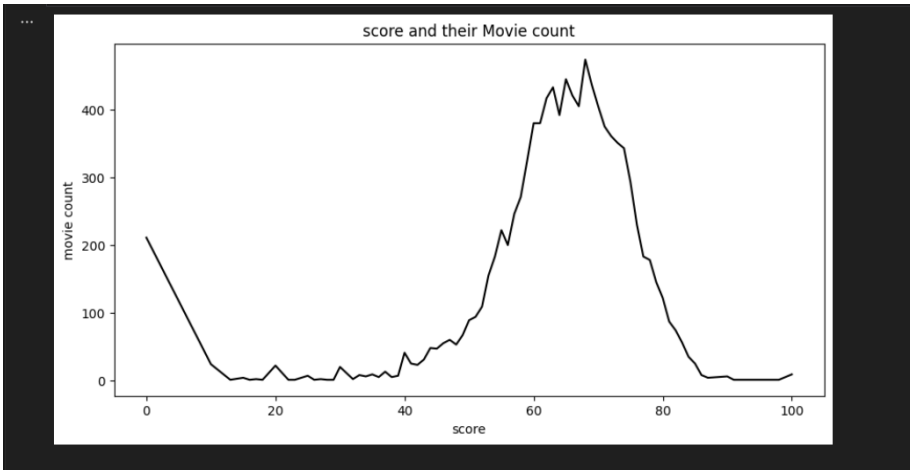


Cette analyse est pertinente car elle nous permet :

- Identifier des Saisonnalités : Détecter des tendances saisonnières dans les sorties de films. Par exemple, constater qu'il y a plus de sorties de films pendant certains mois de l'année.
- Prévisions et Modélisation : En comprenant les tendances de sorties de films par mois, il est possible de développer des modèles de prévision pour estimer le succès futur des films en

fonction de leur date de sortie.

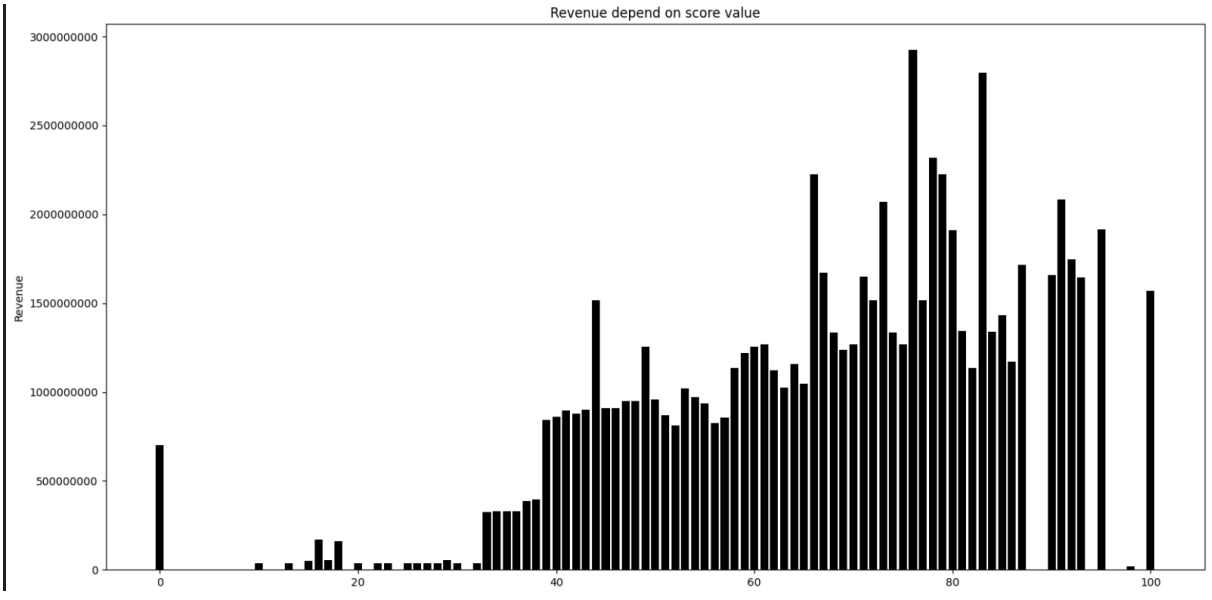
Création d'un diagramme en ligne pour visualiser le nombre de films en fonction de leur score:



Cela permet de comprendre comment les scores des films sont distribués et de détecter des tendances ou des anomalies.

- Identifier la Distribution des Scores : Visualiser la distribution des scores des films aide à comprendre si les films ont tendance à obtenir des scores élevés, moyens ou bas. Cela peut révéler des tendances générales de qualité dans le dataset.
- Détecter des Anomalies : Identifier des anomalies ou des irrégularités, comme un nombre inhabituellement élevé de films avec un score spécifique.
- Segmenter les Films : Cette analyse permet de segmenter les films en fonction de leur score, ce qui peut être utile pour des analyses ultérieures. Par exemple, examiner plus en détail les films avec des scores élevés pour identifier des caractéristiques communes.

Création d'un diagramme à barres pour visualiser la dépendance des revenus par rapport aux scores des films:



Ceci fournit des insights précieux sur la relation entre la qualité perçue d'un film (son score) et son succès commercial (revenu).

- Analyser la Relation entre Score et Revenu : Visualiser comment les revenus varient en fonction des scores permet de comprendre si des films mieux notés tendent à générer plus de revenus.
- Identifier des Tendances : Cela peut révéler des tendances ou des corrélations entre la qualité d'un film et son succès financier, aidant ainsi à identifier les scores qui maximisent les revenus.
- Détecter des Anomalies : Des anomalies comme des films avec de très bons scores mais des revenus faibles (ou vice versa) peuvent être détectées et analysées plus en détail.

4. Prétraitement des données:

Cette étape permet de transformer les données brutes en un format adapté à l'apprentissage automatique.

Description du dataset:

1. Taille du dataset : Notre dataset contient 10178 entrées et 15 colonnes.
2. Colonnes : Les colonnes comprennent des informations telles que le nom, la date, le score, le genre, un résumé, l'équipe de production, le titre original, le statut, la langue d'origine, le budget, les revenus, le pays, le genre principal, l'année de sortie et le mois de sortie.
3. Données manquantes : Certaines colonnes ont des données manquantes. Par exemple, la colonne "genre" a 85 valeurs manquantes et la colonne "primary_genre" en a également 85.
4. Statistiques descriptives :
    - Le score moyen est d'environ 63, avec un minimum de 0 et un maximum de 100.
    - Le budget moyen est d'environ 64,8 millions, avec un minimum de 1 et un maximum de 460 millions.
    - Les revenus moyens sont d'environ 253,1 millions, avec un minimum de 0 et un maximum de 2,9 milliards.
    - Les films couvrent une période allant de 1903 à 2023, avec une sortie plus fréquente dans les années 2000.
    - Les genres sont divers, avec 2303 valeurs uniques.
    - Les langues originales comprennent 54 langues différentes.
    - Les pays de production comprennent 60 pays différents.
5. Diversité : Notre dataset semble couvrir un large éventail de films en termes de genre, de budget, de revenus, de langues et de pays de production.

Feature Engineering :

Créer de nouvelles caractéristiques à partir des données existantes.

```python
# Extracting the primary genre of the movie
df['primary_genre'] = df['genre'].str.split(',').str[0].str.strip()

# Extracting the year and the month when the movie was released
df["Release_year"] = pd.to_datetime(df["date_x"].values).year
df["Release_month"] = pd.to_datetime(df["date_x"].values).month
```

Identification des colonnes caractéristiques et cible à partir du dataset initial:

Suppression des colonnes telles que "names", "date_x", "genre", "overview", etc.,car elles semblent contenir des informations qui ne sont pas pertinentes pour la prédiction des revenus des films. En supprimant ces colonnes, on s'assure que le modèle ne soit pas perturbé par des données inutiles et peut se concentrer sur les caractéristiques qui ont un impact sur la variable cible. Cela aide également à réduire la complexité du modèle et à améliorer sa capacité à généraliser sur de nouvelles données.

	score	budget_x	country	primary_genre	Release_year	Release_month
0	73.0	75000000.0	AU	Drama	2023	3
1	78.0	460000000.0	AU	Science Fiction	2022	12
2	76.0	100000000.0	AU	Animation	2023	4
3	70.0	12300000.0	AU	Animation	2023	1
4	61.0	77000000.0	US	Action	2023	3

Standardisation des valeurs numériques et Encodage one-hot des données catégorielles

```python
# Applying standardization to scale our numerical values
num_pipeline = Pipeline([
    ("imputer", SimpleImputer(strategy="mean")),
    ("scaler", StandardScaler())
])

# Vectorizing our categorical data via OneHotEncoder
cat_pipeline = Pipeline([
    ("imputer", SimpleImputer(strategy="most_frequent")),
    ("encoder", OneHotEncoder(handle_unknown='ignore'))
])
```

- Remplacement des valeurs manquantes pour garantir l'intégrité des données et éviter les biais..
- Encodage des variables catégorielles: transformer ces variables en variables numériques car les algorithmes d'apprentissage automatique nécessitent des entrées numériques.
- Standarisation des valeurs numériques: mettre à l'échelle les données pour que les caractéristiques sur différentes échelles soient comparables.

Partitionnement des données en ensembles d'entraînement et de test:

```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

5. Approche et Algorithmes Utilisés

Pour résoudre notre problématique de prédiction des revenus des films, nous avons utilisé plusieurs algorithmes d'apprentissage supervisé :

1. **Régression linéaire** : Pour établir une relation linéaire entre les caractéristiques et les recettes.
2. **Support Vector Regression** (SVR) : Pour modéliser des relations complexes.

3. **Decision Tree Regressor** : Pour capturer des relations non linéaires.
4. **Random Forest Regressor** : Pour améliorer les performances en combinant plusieurs modèles faibles.

Méthodologie de développement et d'évaluation des modèles:

Pour développer et évaluer nos modèles de régression, nous avons utilisé une approche basée sur des pipelines de machine learning.

Définition du pipeline

Nous avons défini une fonction pipeline qui prend en entrée un préprocesseur et une fonction de modèle, et renvoie un pipeline scikit-learn. Le préprocesseur est appliqué en premier, suivi du modèle. Cette approche nous permet de standardiser le traitement des données en amont du modèle, ce qui est crucial pour obtenir des prédictions précises et fiables.

Évaluation des modèles de régression

Nous avons également défini une fonction evaluate_regression qui calcule les métriques d'évaluation clés pour les modèles de régression : la Mean Absolute Error (MAE), la Mean Squared Error (MSE) et le coefficient de détermination (R-squared). Ces métriques nous permettent d'évaluer la performance de nos modèles et de les comparer entre eux.

Entraînement et prédiction

Enfin, pour entraîner et prédire à l'aide de nos modèles de régression, nous avons utilisé la fonction training_prediction. Cette fonction entraîne le modèle sur les données d'entraînement, effectue des prédictions sur les données de test, puis évalue la performance du modèle à l'aide des métriques définies précédemment.

```
def pipeline(model_func, preprocessor):

    pipeline = Pipeline([
        ("preprocessor", preprocessor),
        ("classifier", model_func())
    ])

    return pipeline

def evaluate_regression(y_true, y_pred):
    mae = mean_absolute_error(y_true, y_pred)
    mse = mean_squared_error(y_true, y_pred)
    r2 = r2_score(y_true, y_pred)

    return {
        "MAE": mae,
        "MSE": mse,
        "R-squared": r2
    }

def training_prediction(X_train, y_train, X_test, y_test, model):

    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    evaluation = evaluate_regression(y_test, y_pred)

    return {
        "prediction": y_pred,
        "evaluation": evaluation,
    }
```

Évaluation des modèles de régression avec Cross-Validation:

Nous avons évalué quatre modèles de régression différents en utilisant la validation croisée à 5 plis. Les modèles évalués sont la Régression Linéaire, le SVR, le Régresseur Arbre de Décision et le Régresseur Forêt Aléatoire.

```
linear_reg_pipe = pipeline(LinearRegression, preprocessor)

svr_pipe = pipeline(SVR, preprocessor)

dt_pipe = pipeline(DecisionTreeRegressor, preprocessor)

rf_pipe = pipeline(RandomForestRegressor, preprocessor)

# Perform k-fold cross-validation
kf = KFold(n_splits=5, shuffle=True)

models = [linear_reg_pipe, svr_pipe, dt_pipe, rf_pipe]
model_names = ['Linear Regression', 'SVR', 'Decision Tree Regressor', 'Random Forest Regressor']

for model, name in zip(models, model_names):
    scores = cross_val_score(model, X_train, y_train, cv=kf, scoring='r2')
    print(f"{name}: Mean r2: {scores.mean()}, Std r2: {scores.std()}")
```

Résultats de la validation croisée (R-squared):

- **Régression Linéaire**: Mean R-squared: 0.543, Std R-squared: 0.016
- **SVR**: Mean R-squared: -0.134, Std R-squared: 0.029
- **Régresseur Arbre de Décision**: Mean R-squared: 0.420, Std R-squared: 0.029
- **Régresseur Forêt Aléatoire**: Mean R-squared: 0.689, Std R-squared: 0.009

Explication des résultats:

- La Régression Linéaire a obtenu un R-squared moyen de 0.543 avec un écart-type de 0.016, indiquant une performance modérée mais relativement stable sur les différentes itérations de la validation croisée.
- Le SVR a donné des résultats surprenants avec un R-squared moyen négatif de -0.134, ce qui suggère que le modèle ne parvient pas à capturer la relation linéaire entre les variables dans nos données. L'écart-type élevé de 0.029 indique également une grande variabilité des performances sur les différents plis.
- Le Régresseur Arbre de Décision a obtenu un R-squared moyen de 0.420 avec un écart-type de 0.029. Bien que ce modèle soit moins performant que la Régression Linéaire, il affiche une performance raisonnable sur nos données.
- Le Régresseur Forêt Aléatoire a obtenu le meilleur R-squared moyen de 0.689 avec un écart-type de 0.009, indiquant une performance élevée et une stabilité relativement bonne sur les différentes itérations de la validation croisée.

⇒ L'évaluation des différents modèles de régression a révélé que le modèle SVR (Support Vector Regression) a présenté des performances inférieures aux autres modèles testés. En conséquence, il a été exclu de l'analyse ultérieure. Alors que le Random Forest Regressor semble être le plus prometteur en termes de performance pour notre tâche de prédiction.

Fine-Tuning

En utilisant la classe GridSearchCV de la bibliothèque Scikit-learn, nous avons pu automatiser la recherche des meilleurs hyperparamètres en ajustant plusieurs modèles d'arbre de décision avec différentes combinaisons d'hyperparamètres .

```
param_grid_dtr = {
    'classifier__max_depth': [3, 5, 7, 10],
    'classifier__min_samples_split': [2, 5, 10],
    'classifier__min_samples_leaf': [1, 2, 4],
    'classifier__max_features': ['auto', 'sqrt', 'log2']
}

grid_search_dtr = GridSearchCV(dt_pipe, param_grid_dtr, cv=5)
grid_search_dtr.fit(X_train, y_train)

# Best parameter found
best_dtr_params = grid_search_dtr.best_params_
print("Best Hyperparameters:", best_dtr_params)

best_dtr = grid_search_dtr.best_estimator_
print("best estimator: ", best_dtr)
```

6. Grille d'évaluation et Comparaison des Algorithmes

Métriques d'évaluation

```
print("Evaluation scores:")

predictions_evaluations = {}

predictions_evaluations["LR"] = training_prediction(X_train, y_train, X_test, y_test, linear_reg_pipe)
print("LR: ", predictions_evaluations["LR"]["evaluation"])

predictions_evaluations["DTR"] = training_prediction(X_train, y_train, X_test, y_test, best_dtr)
print("DTR: ", predictions_evaluations["DTR"]["evaluation"])

predictions_evaluations["RFR"] = training_prediction(X_train, y_train, X_test, y_test, best_rf)
print("RFR: ", predictions_evaluations["RFR"]["evaluation"])
```

Résultats de l'évaluation des modèles de prédiction

Modèle	MAE	MSE	R^2
LR	136,120,103.44	3.89e+16	0.5424
DTR	143,425,980.73	4.68e+16	0.4500
RFR	98,297,897.58	2.47e+16	0.7093

Le modèle de forêt aléatoire (RFR) a obtenu le meilleur score R^2 de 0.7093, indiquant qu'il explique le mieux la variance de la variable dépendante parmi les trois modèles évalués. Le modèle de régression linéaire (LR) a également montré des performances raisonnables avec un R^2 de 0.5424. En revanche, le modèle d'arbre de décision (DTR) a montré des performances inférieures avec un R^2 de 0.4500, indiquant qu'il explique moins de variance que les deux autres modèles.

⇒ Il est clair que le modèle de forêt aléatoire (RFR) se distingue comme le meilleur choix pour ce problème spécifique. Donc c'est celui recommandé pour prédire les revenus des films en raison de ses performances supérieures et de sa capacité à généraliser à de nouvelles données.

Prédictions des revenus pour un nouveau film

```
sample_to_predict = pd.DataFrame([[61.0, 77000000.0 , "US", "Action", 2023, 3]],
                                 columns=["score", "budget_x", "country",  "primary_genre", "Release_year", "Release_month"])

pred_lr = linear_reg_pipe.predict(sample_to_predict)
pred_dtr = best_dtr.predict(sample_to_predict)
pred_rfr = best_rf.predict(sample_to_predict)

print("Prediction for Linear Regression:", pred_lr)
print("Prediction for Decision Tree Regressor:", pred_dtr)
print("Prediction for Random Forest Regressor:", pred_rfr)
```

Les résultats des prédictions sont les suivants :

- Prédiction pour la régression linéaire : 275,933,767 USD
- Prédiction pour l'arbre de décision : 445,828,062 USD
- Prédiction pour la forêt aléatoire : 444,524,763 USD

Ces chiffres représentent les revenus projetés pour le film en fonction des caractéristiques fournies.

Encadré par: Mr Zakaria Haja

Préparé par:

Aït Ben Addi Yassine

Brika Meryeme

Ech-chadli Hamza