Autor: Danielle KOUTON

Project : Diabetes classification Using Machine Learning

Project Requirements

Diabetes dataset: The dataset consists of various medical predictor variables and a target variable (diabetes outcome: 0 or 1). The features include factors like glucose level, blood pressure, BMI, age, etc.

1. Data Preprocessing

```
import pandas as pd
```
✓ 0.0s

```
from pandas import read_csv
diabetes = pd.read_csv('diabetes.csv')
diabetes.head()
```
✓ 0.0s

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

On procède à l'importation du fichier diabetes.csv. On affiche les cinq premiers données du tableau.
Après on vérifie les infos de chaque colonne du dataframe pour une bonne analyse des données.

```
diabetes.info()
✓ 0.0s
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

On vois à travers cette image que les colonnes sont soit de type int, soit de type float. Ce qui est bien pour la suite du traitement.

Après on vérifie s'il y a les données manquantes comme le montre l'image ci_dessous :

```
diabetes.isnull().sum()
✓ 0.0s
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

On constate qu'on a pas de données manquantes dans cette database.

2. Features selection

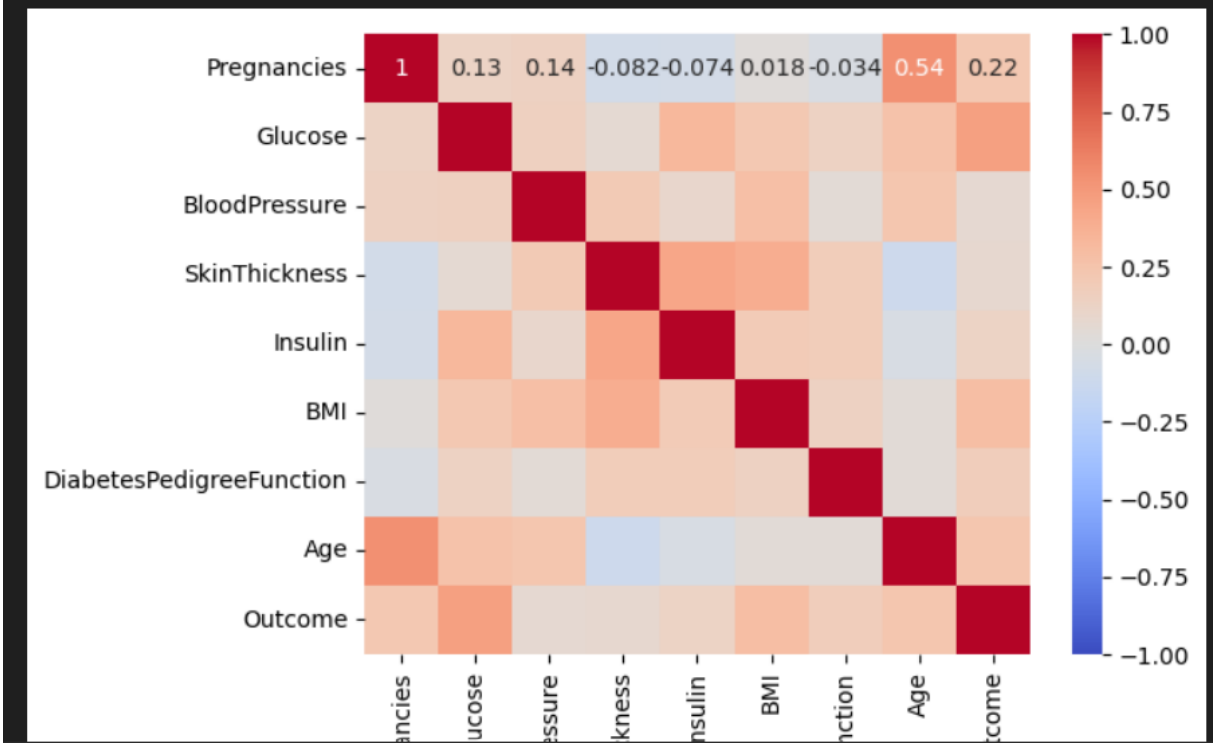
Pour une bonne analyse exploratoire des données on a mis un outil essentiel qui est la matrice de correlation pour mieux comprendre les relations entre les variables indépendantes et la variable cible.

```
import seaborn as sns
import matplotlib.pyplot as plt


correlation_matrix = diabetes.corr()
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", vmin=-1, vmax=1, cbar=True)

plt.show()
```

✓ 0.2s



On a utilisé aussi la librairie seaborn pour une pour une bonne visualisation de la distribution des données numériques comme le montre l'image ci-dessous.

```
import seaborn as sns
import matplotlib.pyplot as plt



sns.pairplot(diabetes)

# Afficher le graphique
plt.show()
```

Après analyse on constate que la relation entre la variable cible et les variables indépendantes est linéaire. On décide alors de réduit l'espace des caractéristiques en utilisant le PCA et de séparer en deux les données(variable cible et variables indépendantes) pour mieux évaluer la performance sur les données non vues.

3. Model Selection

Après analyse de la datasets on constate qu'on a affaire à une datasets dont la varible cible(Outcome) est une variable catégorielle. Outcome est une variable catégorielle car elle est binaire.
Ce qui nous emmène à choisis les modèles de classification qui sont : Logistic Regression, RandomForestClassifier, Neural Networks et le KNN.

4. Model évaluation

Model	Accuracy	Precision	Recall	F1-score
Regression Logistic	0.7338	0.6591	0.5273	0.5859
Neural Network	0.6948	0.6333	0.3455	0.4471
Random Forest	0.6883	0.5593	0.6000	0.5789
KNN	0.6948	0.5741	0.5636	0.5688

Pour mieux apprécié les modèles nous allons nous basé sur les F1-score. On constate ici que les F1-score les plus élevées sont ce de la Regression Logistic et ce du Random Forest. Nous allons donc comparer ces deux models pour pouvoir en décider du meilleur.

- Regression Logistic
  On constate ici que que le recall est de 52,73%. Ce qui signifie qu'il y'a près de la moitié des cas de diabetes qui sont pas détecté. En plus le F1-score est inférieur à la précision. Ce qui nous emmène a dit que le modèle a du mal à trouver tous les instances positives dans les données. Il manque un certains nombre de cas de diabètes.
- Random Forest
  Dans ce cas ci l'accuracy qui est de 68,83% montre que près de 69% des prédictions du modèles qui sont correctes. En plus

la précision est de 55,93% ce qui signifie que près de 56% des prédictions positives sont correctes. On constate aussi que le recall est supérieur au F1-score. Ce qui signifie que le modèl est modérément bon pour identifier les vrais cas positifs.

Conclusion

Au vue des deux analyses on peut déduire que le model le plus approprié dans ce cas est le random Forest car sa capacité à détecter les vrais cas positifs est supérieurs à celle de la Logistic Regression.

5. Hyperparameter Turning

   On constate que quand bien même le model Random Forest à une capacité de prédiction élevée il y a des problèmes compte tenu des faibles résultats obtenus. Pour alors avoir des meilleurs résultats on a donc utilisés le Grid Search. On a opté pour le Grid Search car elle permet d'obtenir la meilleure combinaison d'hyperparamètres.

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier


rf_model = RandomForestClassifier(random_state=42)

# Définir la grille de paramètres
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [5, 10, 15],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False]
}

# Configurer le GridSearchCV
grid_search = GridSearchCV(estimator=rf_model, param_grid=param_grid,
                           cv=5, n_jobs=-1, verbose=2, scoring='accuracy')

# Effectuer la recherche
grid_search.fit(X_train, y_train)

# Meilleurs paramètres
print("Best parameters found: ", grid_search.best_params_)
print("Best accuracy: ", grid_search.best_score_)
```

```
✓ 45.8s
Fitting 5 folds for each of 162 candidates, totalling 810 fits
Best parameters found:  {'bootstrap': True, 'max_depth': 5, 'min_samples_leaf': 4, 'min_samples_split': 2, 'n_estimators': 100}
Best accuracy:  0.7361721977875517
```

   Comme le montre l'image ci-dessus on constate que après la combinaison d'hyperparamètres le modèl Random Forest donne de meilleure performance avec une accuracy de 73,62%.

   Ce qui est largement meilleure que la précédente.

Conclusion

Après plusieurs analyse on peut dire que le model qui peut nous aidé à avoir de précisons sur les vrais cas de diabètes positives est le model Random Forest car son accuracy est de 73,62%. Aussi son recall qui est sa capacité a détecté les vrais cas de diabètes est élevée. Il est donc le modèles le plus approprié et le plus performant.

Comptes tenu des résultats de ce model on peut dire qu'il y a beaucoup de cas de diabetes positives.