

Import Libraries


```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, roc_auc_score, roc_curve
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from xgboost import XGBClassifier
import gradio as gr
```

Loading the data

In this project, the dataset was imported from the open source "KAGGLE" and saved as a CSV file named Hiring.csv. It ontains a collection of common student data

```
df=pd.read_csv("/hiring.csv.zip")
```

```
df.head()
```




	Age	Gender	EducationLevel	ExperienceYears	PreviousCompanies	DistanceFromCompany	InterviewScore	SkillScore	PersonalityScore	Re
0	26	1	2	0	3	26.783828	48	78	91	
1	39	1	4	12	3	25.862694	35	68	80	
2	48	0	2	3	2	9.920805	20	67	13	
3	34	1	2	5	2	6.407751	36	27	70	
4	30	0	1	6	1	43.105343	23	52	85	

Next steps: [Generate code with df](#) [View recommended plots](#) [New interactive sheet](#)

Exploratory of data analysis : The dataset contains hiring.csv contains student information and their corresponding performances.We load the data into a data frame to understand its stucture and content

```
df.info()
```



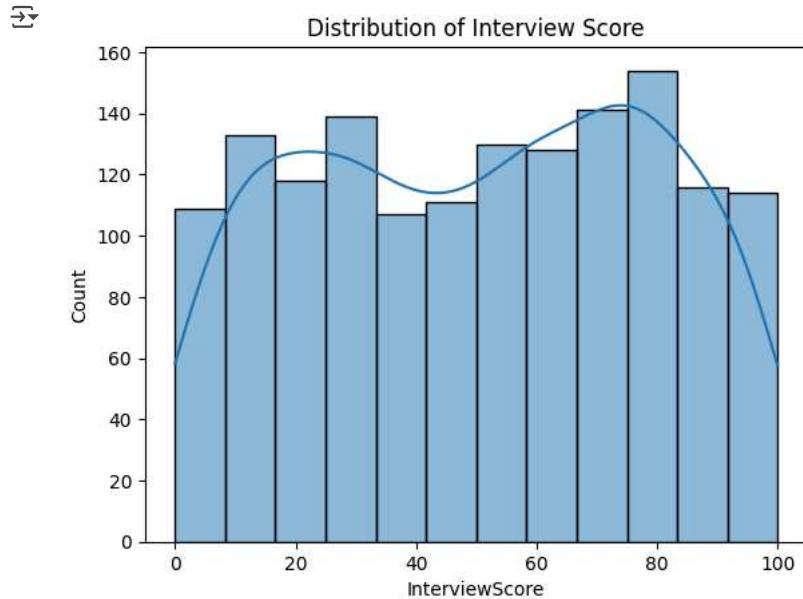
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1500 entries, 0 to 1499
Data columns (total 11 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Age                   1500 non-null  int64
1   Gender                1500 non-null  int64
2   EducationLevel        1500 non-null  int64
3   ExperienceYears       1500 non-null  int64
4   PreviousCompanies    1500 non-null  int64
5   DistanceFromCompany  1500 non-null  float64
6   InterviewScore       1500 non-null  int64
7   SkillScore           1500 non-null  int64
8   PersonalityScore     1500 non-null  int64
9   RecruitmentStrategy  1500 non-null  int64
10  HiringDecision       1500 non-null  int64
dtypes: float64(1), int64(10)
memory usage: 129.0 KB
```

Visualising dashboards: A visualization dashboard is an interactive interface that presents data visually using charts, graphs, and tables. It allows users to monitor, explore, and analyze key metrics at a glance.

Purpose of Visualization Dashboards :

- Summarization: Aggregate large datasets into digestible views.
- Analysis: Identify trends, outliers, and correlations.
- Monitoring: Track performance metrics in real time.
- Storytelling: Present insights clearly to stakeholders.

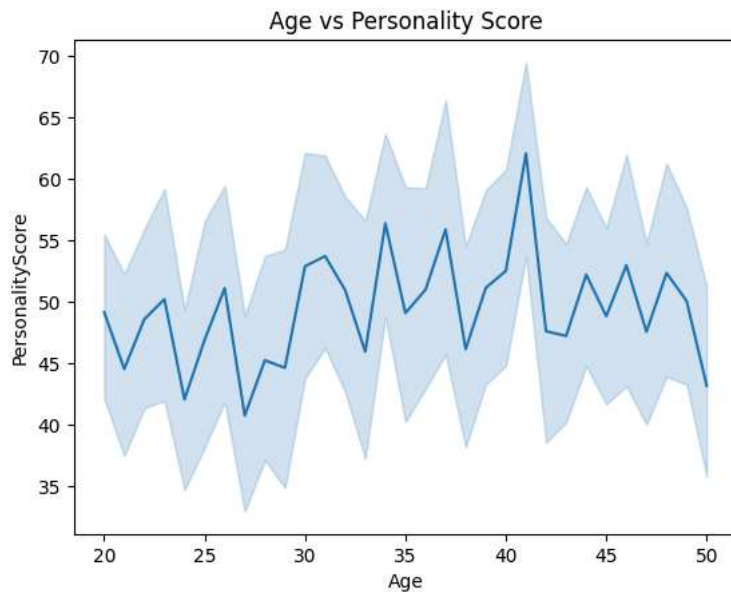
```
sns.histplot(df['InterviewScore'], kde=True)
plt.title('Distribution of Interview Score')
plt.show()
```



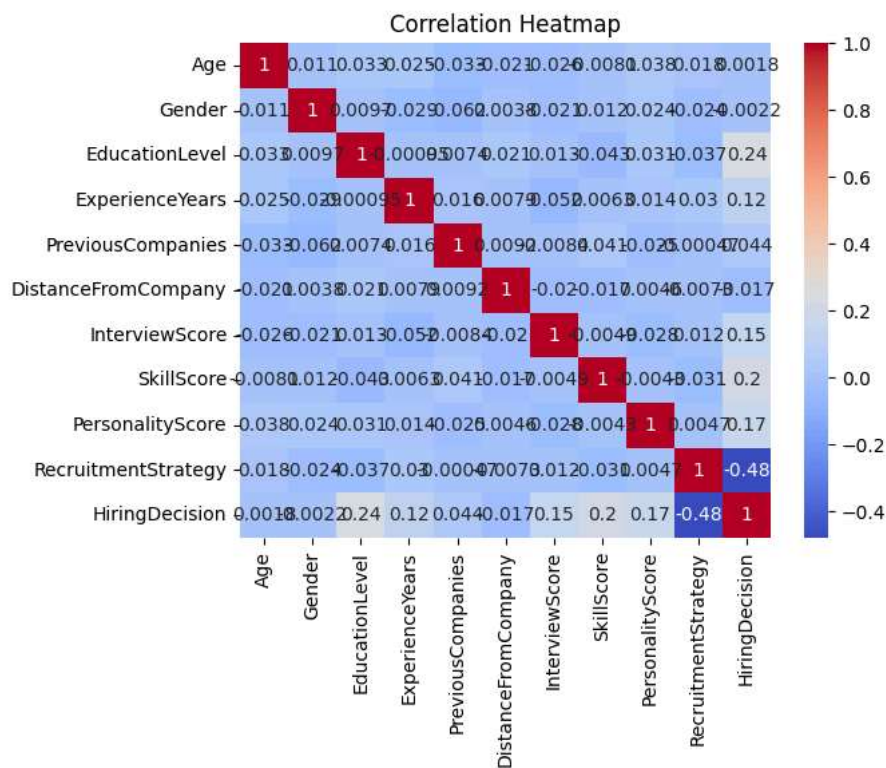
```
sns.boxplot(x='HiringDecision', y='SkillScore', data=df)
plt.title('Skill Score by Hiring Decision')
plt.show()
```



```
sns.lineplot(x='Age', y='PersonalityScore', data=df)
plt.title('Age vs Personality Score')
plt.show()
```



```
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



Feature Engineering : The dataset contains information about candidates and their hiring decisions. Here's a breakdown of the feature engineering process that can be applied to this dataset:

Feature Engineering Ideas

1. Binning/Grouping Continuous Variables

- Age Group: Bucket Age into categories (e.g., <30, 30–40, >40).
- Experience Level: Convert ExperienceYears into categorical levels (e.g., Junior, Mid, Senior).

2. Interaction Features

- Total Assessment Score: Combine InterviewScore + SkillScore + PersonalityScore to form a holistic metric.

- Efficiency Ratio: $\text{SkillScore} / \text{ExperienceYears}$ (handles how quickly someone has gained skill).

3. Normalization / Scaling

- Scale numeric values like `DistanceFromCompany`, `InterviewScore`, etc., especially if using distance-based algorithms like KNN or SVM.

4. Encoding Categorical Variables

- Gender, EducationLevel, and RecruitmentStrategy may need one-hot encoding or ordinal encoding depending on the model.

5. Missing Data Handling

- Check for and handle missing values (imputation or dropping rows).

6. Outlier Detection

- Identify unusual values in `DistanceFromCompany`, `SkillScore`, etc., and treat them appropriately.

7. Custom Domain Features

- Loyalty Indicator: $\text{ExperienceYears} / \text{PreviousCompanies}$ — a higher ratio might suggest candidate stability.
- Commute Stress Factor: Combine `DistanceFromCompany` and `PersonalityScore` (long commute + low personality resilience could be a concern).

```
df.dropna(inplace=True)
df = df[df['PreviousCompanies'] > 0] # avoid division by zero

# Create new features
df['InterviewToSkillRatio'] = df['InterviewScore'] / (df['SkillScore'] + 1)
df['ExperiencePerCompany'] = df['ExperienceYears'] / df['PreviousCompanies']

# Separate features and target
X = df.drop("HiringDecision", axis=1)
y = df["HiringDecision"]

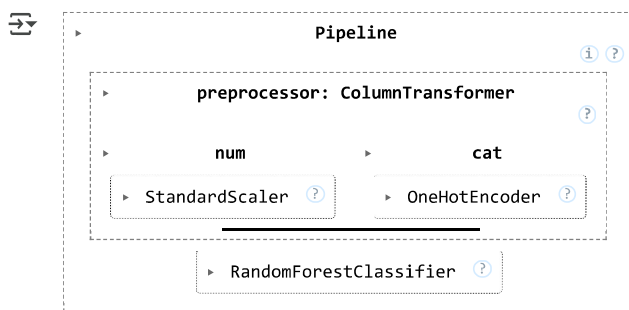
# Categorical and numerical columns
categorical = ['RecruitmentStrategy']
numerical = X.select_dtypes(include=[np.number]).columns.difference(categorical)

preprocessor = ColumnTransformer([
    ('num', StandardScaler(), numerical),
    ('cat', OneHotEncoder(drop='first'), categorical)
])
```

Train the model: In the training and the Testing module we split the data into testing and training phases by using the RandomForest Classifier

```
pipe = Pipeline([
    ('preprocessor', preprocessor),
    ('clf', RandomForestClassifier(random_state=42))
])

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
pipe.fit(X_train, y_train)
```



Model Evaluation: Evaluate performance using metrics:

- Accuracy: Overall correctness
- Precision / Recall / F1 Score: Better for imbalanced data

- Confusion Matrix: Shows true vs. predicted
- ROC-AUC: Evaluates performance at all thresholds

```
from sklearn.metrics import classification_report, confusion_matrix
```

```
y_pred = pipe.predict(X_test)
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

```

↔
      precision    recall  f1-score   support

     0       0.94      0.98      0.96       215
     1       0.93      0.84      0.88        85

 accuracy          0.94
 macro avg       0.94      0.91      0.92
 weighted avg    0.94      0.94      0.94

[[210   5]
 [ 14  71]]

```

Double-click (or enter) to edit

```

def predict_hiring(Age, Gender, EducationLevel, ExperienceYears, PreviousCompanies, DistanceFromCompany, InterviewScore, SkillScore, PersonalityScore):
    df_input = pd.DataFrame([[Age, Gender, EducationLevel, ExperienceYears, PreviousCompanies, DistanceFromCompany, InterviewScore, SkillScore, PersonalityScore]])
    df_input['InterviewToSkillRatio'] = df_input['InterviewScore'] / (df_input['SkillScore'] + 1)
    df_input['ExperiencePerCompany'] = df_input['ExperienceYears'] / df_input['PreviousCompanies']
    prediction = pipe.predict(df_input)[0]
    return "Hired" if prediction == 1 else "Not Hired"

inputs = [
    gr.Number(label="Age"),
    gr.Number(label="Gender (0=Male, 1=Female)"),
    gr.Number(label="EducationLevel (1=Bach1, 2=Bach2, 3=Master, 4=PhD)"),
    gr.Number(label="ExperienceYears"),
    gr.Number(label="PreviousCompanies"),
    gr.Number(label="DistanceFromCompany"),
    gr.Number(label="InterviewScore"),
    gr.Number(label="SkillScore"),
    gr.Number(label="PersonalityScore"),
    gr.Number(label="RecruitmentStrategy (1=Agressive, 2=Moderate, 3=Conservative)")
]
app = gr.Interface(fn=predict_hiring, inputs=inputs, outputs="text", title="Hiring Decision Predictor")

app.launch()

```

↗ It looks like you are running Gradio on a hosted a Jupyter notebook. For the Gradio app to work, sharing must be enabled. Automatically Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
 * Running on public URL: <https://ba4f98891782622c7a.gradio.live>

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working dir

ExperienceYears

0

PreviousCompanies

0

Final version:

In the final version, all components of the Hiring data were integrated into a complete and functional system. This version includes data handling, semantic similarity scoring, experience evaluation, and a user-friendly interactive interface. The classification evaluates each response, provides scores, and offers constructive recruitment strategy, simulating a real working of the Hiring process experience. This finalized version is ready for demonstration, testing, and further enhancement based on future requirements.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
import gradio as gr

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.ensemble import RandomForestClassifier

warnings.filterwarnings("ignore")

df = pd.read_csv("/hiring.csv.zip")

print(df.head())
print(df.info())
print(df.describe())

sns.histplot(df['InterviewScore'], kde=True)
plt.title('Distribution of Interview Score')
plt.show()

sns.boxplot(x='HiringDecision', y='SkillScore', data=df)
plt.title('Skill Score by Hiring Decision')
plt.show()

sns.lineplot(x='Age', y='PersonalityScore', data=df)
plt.title('Age vs Personality Score')
```

```

plt.figure(figsize=(10, 10))
plt.show()

sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()

df.dropna(inplace=True)
df = df[df['PreviousCompanies'] > 0]

df['InterviewToSkillRatio'] = df['InterviewScore'] / (df['SkillScore'] + 1)
df['ExperiencePerCompany'] = df['ExperienceYears'] / df['PreviousCompanies']
X = df.drop("HiringDecision", axis=1)
y = df["HiringDecision"]

categorical = ['RecruitmentStrategy']
numerical = X.select_dtypes(include=[np.number]).columns.difference(categorical)

preprocessor = ColumnTransformer([
    ('num', StandardScaler(), numerical),
    ('cat', OneHotEncoder(drop='first'), categorical)
])

pipe = Pipeline([
    ('preprocessor', preprocessor),
    ('clf', RandomForestClassifier(random_state=42))
])

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
pipe.fit(X_train, y_train)

y_pred = pipe.predict(X_test)
print("\n--- Evaluation Metrics ---")
print("Accuracy:", accuracy_score(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))

def predict_hiring(Age, Gender, EducationLevel, ExperienceYears, PreviousCompanies,
                   DistanceFromCompany, InterviewScore, SkillScore, PersonalityScore, RecruitmentStrategy):

    if Age < 18:
        return " Age must be at least 18"
    if not (1 <= EducationLevel <= 4):
        return " EducationLevel must be between 1 and 4"
    if ExperienceYears < 0:
        return " Experience must be 0 or more"
    if PreviousCompanies < 0:
        return " Previous Companies must be 0 or more"
    if DistanceFromCompany >= 15:
        return " Distance must be less than 15 km"
    if InterviewScore < 70:
        return " Interview Score must be at least 70"
    if SkillScore < 70:
        return " Skill Score must be at least 70"
    if PersonalityScore < 60:
        return " Personality Score must be at least 60"
    if not (1 <= RecruitmentStrategy <= 3):
        return " Recruitment Strategy must be between 1 and 3"

df_input = pd.DataFrame([[Age, Gender, EducationLevel, ExperienceYears, PreviousCompanies,
                           DistanceFromCompany, InterviewScore, SkillScore, PersonalityScore, RecruitmentStrategy]],
                          columns=['Age', 'Gender', 'EducationLevel', 'ExperienceYears',
                                   'PreviousCompanies', 'DistanceFromCompany', 'InterviewScore',
                                   'SkillScore', 'PersonalityScore', 'RecruitmentStrategy'])

df_input['InterviewToSkillRatio'] = df_input['InterviewScore'] / (df_input['SkillScore'] + 1)
df_input['ExperiencePerCompany'] = df_input['ExperienceYears'] / (df_input['PreviousCompanies'] + 1)

prediction = pipe.predict(df_input)[0]
return "Not Hired" if prediction == 1 else " Hired"

inputs = [
    gr.Number(label="Age (min 18)", minimum=18),
    gr.Number(label="Gender (0=Male, 1=Female)"),
    gr.Number(label="Education Level (1=Bach1, 2=Bach2, 3=Master, 4=PhD)", minimum=1, maximum=4),

```

```
gr.Number(label="Experience Years (min 0)", minimum=0),
gr.Number(label="Previous Companies (min 0)", minimum=0),
gr.Number(label="Distance From Company (must be <15)", maximum=14.99),
gr.Number(label="Interview Score (min 70)", minimum=70),
gr.Number(label="Skill Score (min 70)", minimum=70),
gr.Number(label="Personality Score (min 60)", minimum=60),
gr.Number(label="Recruitment Strategy (1=Agressive, 2=Moderate, 3=Conservative)", minimum=1, maximum=3)
]
app = gr.Interface(
    fn=predict_hiring,
    inputs=inputs,
    outputs="text",
    title="Hiring Decision Predictor",
    description="Enter candidate data. The model will predict if the candidate will be hired or not. Validation and margins enforced."
)

app.launch()
```



```

↩
  Age  Gender  EducationLevel  ExperienceYears  PreviousCompanies  \
0    26      1              2              0              3
1    39      1              4             12              3
2    48      0              2              3              2
3    34      1              2              5              2
4    30      0              1              6              1

```

```

  DistanceFromCompany  InterviewScore  SkillScore  PersonalityScore  \
0          26.783828             48           78             91
1          25.862694             35           68             80
2           9.920805             20           67             13
3          6.407751             36           27             70
4          43.105343             23           52             85

```

```

  RecruitmentStrategy  HiringDecision
0              1              1
1              2              1
2              2              0
3              3              0
4              2              0

```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 1500 entries, 0 to 1499
```

```
Data columns (total 11 columns):
```

```

#   Column                Non-Null Count  Dtype
---  -
0   Age                  1500 non-null    int64
1   Gender                1500 non-null    int64
2   EducationLevel        1500 non-null    int64
3   ExperienceYears        1500 non-null    int64
4   PreviousCompanies     1500 non-null    int64
5   DistanceFromCompany    1500 non-null    float64
6   InterviewScore         1500 non-null    int64
7   SkillScore             1500 non-null    int64
8   PersonalityScore       1500 non-null    int64
9   RecruitmentStrategy    1500 non-null    int64
10  HiringDecision        1500 non-null    int64

```

```
dtypes: float64(1), int64(10)
```

```
memory usage: 129.0 KB
```

```
None
```

```

      Age  Gender  EducationLevel  ExperienceYears  \
count  1500.000000  1500.000000  1500.000000  1500.000000
mean    35.148667    0.492000    2.188000    7.694000
std     9.252728    0.500103    0.862449    4.641414
min     20.000000    0.000000    1.000000    0.000000
25%     27.000000    0.000000    2.000000    4.000000
50%     35.000000    0.000000    2.000000    8.000000
75%     43.000000    1.000000    3.000000   12.000000
max     50.000000    1.000000    4.000000   15.000000

```

```

  PreviousCompanies  DistanceFromCompany  InterviewScore  SkillScore  \
count          1500.00000          1500.000000  1500.000000  1500.000000
mean              3.00200           25.505379    50.564000   51.116000
std              1.41067           14.567151    28.626215   29.353563
min              1.00000           1.031376     0.000000    0.000000
25%              2.00000           12.838851    25.000000   25.750000
50%              3.00000           25.502239    52.000000   53.000000
75%              4.00000           37.737996    75.000000   76.000000
max              5.00000           50.992462   100.000000  100.000000

```

```

  PersonalityScore  RecruitmentStrategy  HiringDecision
count          1500.000000          1500.000000  1500.000000
mean             49.387333             1.893333    0.310000
std             29.353201             0.689642    0.462647
min              0.000000             1.000000    0.000000
25%             23.000000             1.000000    0.000000
50%             49.000000             2.000000    0.000000
75%             76.000000             2.000000    1.000000
max            100.000000             3.000000    1.000000

```

