ml-model-comparision

March 16, 2025

[3]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import OneHotEncoder
import warnings
import time
warnings.filterwarnings("ignore")
data = pd.read_csv('C:\\Users\\Manas\\OneDrive\\Documents\\TYDS Final Year␣
 ↪Project\\TYDS Project Data\\Merged_data2.csv')

df = pd.DataFrame(data)
X = df[['combined_skills', 'average_salary_value', 'average_experience']]
y = df['job_title']

X['combined_skills'] = X['combined_skills'].apply(lambda x: ' '.join(x))
X = pd.concat([X.drop('combined_skills', axis=1), X['combined_skills'].str.
 ↪get_dummies(sep=' ')], axis=1)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)
model = RandomForestClassifier()

start_t_fit = time.time()
model.fit(X_train, y_train)
end_t_fit = time.time()
t1_RF1=(end_t_fit-start_t_fit)
print(f"Time taken to fit the data in model[s]: {(end_t_fit-start_t_fit):.3f}")

start_t_pred = time.time()
y_pred = model.predict(X_test)
end_t_pred = time.time()
t2_RF1=(end_t_pred-start_t_pred)
print(f"Time taken to predict[s]: {(end_t_pred-start_t_pred):.3f}")

T_RF1=t1_RF1+t2_RF1
print(f"Total Time:{(T_RF1):.3f}")
```

```
accuracy1 = model.score(X_test, y_test)*100
print('Accuracy:', accuracy1)
```

```
Time taken to fit the data in model[s]: 6.346
Time taken to predict[s]: 1.691
Total Time:8.037
Accuracy: 99.06281032770605
```

[4]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import OneHotEncoder
import warnings
warnings.filterwarnings("ignore")

data = pd.read_csv('C:\\Users\\Manas\\OneDrive\\Documents\\TYDS Final Year␣
 ↪Project\\TYDS Project Data\\Merged_data2.csv')
df = pd.DataFrame(data)
df['skill_count'] = df['combined_skills'].apply(len)

X = df[['combined_skills', 'average_salary_value',␣
 ↪'average_experience','skill_count']]
y = df['job_title']

X['combined_skills'] = X['combined_skills'].apply(lambda x: ' '.join(x))
X = pd.concat([X.drop('combined_skills', axis=1), X['combined_skills'].str.
 ↪get_dummies(sep=' ')], axis=1)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)
model = RandomForestClassifier()
start_t_fit = time.time()
model.fit(X_train, y_train)
end_t_fit = time.time()
t1_RF2=(end_t_fit-start_t_fit)
print(f"Time taken to fit the data in model[s]: {(end_t_fit-start_t_fit):.3f}")
start_t_pred = time.time()
y_pred = model.predict(X_test)
end_t_pred = time.time()
t2_RF2=(end_t_pred-start_t_pred)
print(f"Time taken to predict[s]: {(end_t_pred-start_t_pred):.3f}")

T_RF2=t1_RF2+t2_RF2
print(f"Total Time:{(T_RF2):.3f}")

accuracy2 = model.score(X_test, y_test)*100
print('Accuracy:', accuracy2)
```

```
Time taken to fit the data in model[s]: 6.020
Time taken to predict[s]: 1.584
Total Time:7.603
Accuracy: 99.07522343594836
```

[5]:
```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import StandardScaler
import warnings
warnings.filterwarnings("ignore")

data = pd.read_csv('C:\\Users\\Manas\\OneDrive\\Documents\\TYDS Final Year␣
 ↪Project\\TYDS Project Data\\Merged_data2.csv')

df = pd.DataFrame(data)
df['skill_count'] = df['combined_skills'].apply(len)

scaler = StandardScaler()
df['salary_normalized'] = scaler.fit_transform(df[['average_salary_value']])

X = df[['combined_skills', 'salary_normalized',␣
 ↪'average_experience','skill_count']]
y = df['job_title']

X['combined_skills'] = X['combined_skills'].apply(lambda x: ' '.join(x))
X = pd.concat([X.drop('combined_skills', axis=1), X['combined_skills'].str.
 ↪get_dummies(sep=' ')], axis=1)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
model = RandomForestClassifier()
start_t_fit = time.time()
model.fit(X_train, y_train)
end_t_fit = time.time()
t1_RF3=(end_t_fit-start_t_fit)
print(f"Time taken to fit the data in model[s]: {(end_t_fit-start_t_fit):.3f}")

start_t_pred = time.time()
y_pred = model.predict(X_test)
end_t_pred = time.time()
t2_RF3=(end_t_pred-start_t_pred)
print(f"Time taken to predict[s]: {(end_t_pred-start_t_pred):.3f}")

T_RF3=t1_RF3+t2_RF3
print(f"Total Time:{(T_RF3):.3f}")
```

```
accuracy3 = model.score(X_test, y_test)*100
print('Accuracy:', accuracy3)
```

```
Time taken to fit the data in model[s]: 6.671
Time taken to predict[s]: 1.117
Total Time:7.788
Accuracy: 99.1093594836147
```

[6]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, accuracy_score
import warnings
warnings.filterwarnings("ignore")

data = pd.read_csv('C:\\Users\\Manas\\OneDrive\\Documents\\TYDS Final Year␣
 ↪Project\\TYDS Project Data\\Merged_data2.csv')
df = pd.DataFrame(data)
df['skill_count'] = df['combined_skills'].apply(len)

scaler = StandardScaler()
df['salary_normalized'] = scaler.fit_transform(df[['average_salary_value']])

X = df[['combined_skills', 'salary_normalized',␣
 ↪'average_experience','skill_count']]
y = df['job_title']

X['combined_skills'] = X['combined_skills'].apply(lambda x: ' '.join(x))
X = pd.concat([X.drop('combined_skills', axis=1), X['combined_skills'].str.
 ↪get_dummies(sep=' ')], axis=1)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)

model = DecisionTreeClassifier()
start_t_fit = time.time()
model.fit(X_train, y_train)
end_t_fit = time.time()
t1_DT=(end_t_fit-start_t_fit)
print(f"Time taken to fit the data in model[s]: {(end_t_fit-start_t_fit):.3f}")

start_t_pred = time.time()
y_pred = model.predict(X_test)
end_t_pred = time.time()
```

```
t2_DT=(end_t_pred-start_t_pred)
print(f"Time taken to predict[s]: {(end_t_pred-start_t_pred):.3f}")

T_DT=t1_DT+t2_DT
print(f"Total Time:{(T_DT):.3f}")

accuracy4 = model.score(X_test, y_test)*100
print('Accuracy:', accuracy4)
```

```
Time taken to fit the data in model[s]: 0.375
Time taken to predict[s]: 0.041
Total Time:0.416
Accuracy: 98.81454816285998
```

[7]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")

data = pd.read_csv('C:\\Users\\Manas\\OneDrive\\Documents\\TYDS Final Year␣
 ↪Project\\TYDS Project Data\\Merged_data2.csv')
df = pd.DataFrame(data)
df['skill_count'] = df['combined_skills'].apply(len)

scaler = StandardScaler()
df['salary_normalized'] = scaler.fit_transform(df[['average_salary_value']])

X = df[['combined_skills', 'salary_normalized',␣
 ↪'average_experience','skill_count']]
y = df['job_title']

X['combined_skills'] = X['combined_skills'].apply(lambda x: ' '.join(x))
X = pd.concat([X.drop('combined_skills', axis=1), X['combined_skills'].str.
 ↪get_dummies(sep=' ')], axis=1)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

model = KNeighborsClassifier(n_neighbors=7)
start_t_fit = time.time()
model.fit(X_train, y_train)
end_t_fit = time.time()
t1_KNN=(end_t_fit-start_t_fit)
```

```
print(f"Time taken to fit the data in model[s]: {(end_t_fit-start_t_fit):.3f}")

start_t_pred = time.time()
y_pred = model.predict(X_test)
end_t_pred = time.time()
t2_KNN=(end_t_pred-start_t_pred)
print(f"Time taken to predict[s]: {(end_t_pred-start_t_pred):.3f}")

T_KNN=t1_KNN+t2_KNN
print(f"Total Time:{(T_KNN):.3f}")

accuracy5 = model.score(X_test, y_test)*100
print('Accuracy:', accuracy5)
```

```
Time taken to fit the data in model[s]: 0.209
Time taken to predict[s]: 12.697
Total Time:12.906
Accuracy: 98.80523833167825
```

[8]:
```
import pandas as pd
import time
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
import warnings
warnings.filterwarnings("ignore")

data = pd.read_csv('C:\\Users\\Manas\\OneDrive\\Documents\\TYDS Final Year␣
 ↪Project\\TYDS Project Data\\Merged_data2.csv')
df = pd.DataFrame(data)
df['skill_count'] = df['combined_skills'].apply(len)

scaler = StandardScaler()
df['salary_normalized'] = scaler.fit_transform(df[['average_salary_value']])

X = df[['combined_skills', 'salary_normalized', 'average_experience',␣
 ↪'skill_count']]
y = df['job_title']

X['combined_skills'] = X['combined_skills'].apply(lambda x: ' '.join(x))
X = pd.concat([X.drop('combined_skills', axis=1), X['combined_skills'].str.
 ↪get_dummies(sep=' ')], axis=1)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

model = SVC(kernel='rbf')
```

```
start_t_fit = time.time()
model.fit(X_train, y_train)
end_t_fit = time.time()
t1_SVM = end_t_fit - start_t_fit
print(f"Time taken to fit the SVM model[s]: {t1_SVM:.3f}")

start_t_pred = time.time()
y_pred = model.predict(X_test)
end_t_pred = time.time()
t2_SVM = end_t_pred - start_t_pred
print(f"Time taken to predict using SVM[s]: {t2_SVM:.3f}")

T_SVM = t1_SVM + t2_SVM
print(f"Total Time for SVM: {T_SVM:.3f}")
accuracy6 = model.score(X_test, y_test) * 100
print("SVM Accuracy:", accuracy6)
```

```
Time taken to fit the SVM model[s]: 945.825
Time taken to predict using SVM[s]: 70.698
Total Time for SVM: 1016.522
SVM Accuracy: 98.43284508440914
```

[9]:
```
import pandas as pd
import time
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.preprocessing import StandardScaler
import warnings
warnings.filterwarnings("ignore")

data = pd.read_csv('C:\\Users\\Manas\\OneDrive\\Documents\\TYDS Final Year␣
 ↪Project\\TYDS Project Data\\Merged_data2.csv')
df = pd.DataFrame(data)
df['skill_count'] = df['combined_skills'].apply(len)

scaler = StandardScaler()
df['salary_normalized'] = scaler.fit_transform(df[['average_salary_value']])

X = df[['combined_skills', 'salary_normalized', 'average_experience',␣
 ↪'skill_count']]
y = df['job_title']

X['combined_skills'] = X['combined_skills'].apply(lambda x: ' '.join(x))
X = pd.concat([X.drop('combined_skills', axis=1), X['combined_skills'].str.
 ↪get_dummies(sep=' ')], axis=1)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

model = GaussianNB()
start_t_fit = time.time()
model.fit(X_train, y_train)
end_t_fit = time.time()
t1_NB = end_t_fit - start_t_fit
print(f"Time taken to fit the GaussianNB model[s]: {t1_NB:.3f}")

start_t_pred = time.time()
y_pred = model.predict(X_test)
end_t_pred = time.time()
t2_NB = end_t_pred - start_t_pred
print(f"Time taken to predict using GaussianNB[s]: {t2_NB:.3f}")

T_NB = t1_NB + t2_NB
print(f"Total Time for GaussianNB: {T_NB:.3f}")
accuracy7 = model.score(X_test, y_test) * 100
print("GaussianNB Accuracy:", accuracy7)
```

```
Time taken to fit the GaussianNB model[s]: 1.235
Time taken to predict using GaussianNB[s]: 1.389
Total Time for GaussianNB: 2.624
GaussianNB Accuracy: 98.37077954319761
```

[10]:
```
import pandas as pd
import time
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.preprocessing import StandardScaler
import warnings
warnings.filterwarnings("ignore")

data = pd.read_csv('C:\\Users\\Manas\\OneDrive\\Documents\\TYDS Final Year␣
 ↪Project\\TYDS Project Data\\Merged_data2.csv')
df = pd.DataFrame(data)
df['skill_count'] = df['combined_skills'].apply(len)

scaler = StandardScaler()
df['salary_normalized'] = scaler.fit_transform(df[['average_salary_value']])

X = df[['combined_skills', 'salary_normalized', 'average_experience',␣
 ↪'skill_count']]
y = df['job_title']

X['combined_skills'] = X['combined_skills'].apply(lambda x: ' '.join(x))
```

```
X = pd.concat([X.drop('combined_skills', axis=1), X['combined_skills'].str.
 ↪get_dummies(sep=' ')], axis=1)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

model = MLPClassifier(hidden_layer_sizes=(100,), max_iter=500)
start_t_fit = time.time()
model.fit(X_train, y_train)
end_t_fit = time.time()
t1_MLP = end_t_fit - start_t_fit
print(f"Time taken to fit the MLPClassifier model[s]: {t1_MLP:.3f}")

start_t_pred = time.time()
y_pred = model.predict(X_test)
end_t_pred = time.time()
t2_MLP = end_t_pred - start_t_pred
print(f"Time taken to predict using MLPClassifier[s]: {t2_MLP:.3f}")

T_MLP = t1_MLP + t2_MLP
print(f"Total Time for MLPClassifier: {T_MLP:.3f}")
accuracy8 = model.score(X_test, y_test) * 100
print("MLPClassifier Accuracy:", accuracy8)
```

```
Time taken to fit the MLPClassifier model[s]: 596.629
Time taken to predict using MLPClassifier[s]: 0.260
Total Time for MLPClassifier: 596.890
MLPClassifier Accuracy: 98.92005958291956
```

[11]:
```
temp={
    "Random Forest":[t1_RF1,t2_RF1,T_RF1,accuracy1],
    "Random Forest(Fe_cre)":[t1_RF2,t2_RF2,T_RF2,accuracy2],
    "Random Forest(Scaled)":[t1_RF3,t2_RF3,T_RF3,accuracy3],
    "Decision Tree":[t1_DT,t2_DT,T_DT,accuracy4],
    "KNN Classifier":[t1_KNN,t2_KNN,T_KNN,accuracy5],
    "SVM":[t1_SVM,t2_SVM,T_SVM,accuracy6],
    "GaussianNB":[t1_NB,t2_NB,T_NB,accuracy7],
    "MLPClassifier":[t1_MLP,t2_MLP,T_MLP,accuracy8]
}

Model_Comparision=pd.DataFrame(temp)
Model_Comparision.index = ['T_Fit', 'T_Predict', 'To_Time', 'Accuracy']
pd.set_option('display.width', 1000)
pd.set_option('display.max_columns', None)

print(Model_Comparision)
```

```
           Random Forest  Random Forest(Fe_cre)  Random Forest(Scaled)  Decision
Tree  KNN Classifier          SVM  GaussianNB  MLPClassifier
T_Fit           6.346001               6.019567               6.670857
0.375006        0.208920   945.824725    1.235327     596.629354
T_Predict       1.690826               1.583891               1.117004
0.040999       12.697035    70.697570    1.389002       0.260412
To_Time         8.036827               7.603458               7.787861
0.416005       12.905955  1016.522295    2.624328     596.889766
Accuracy       99.062810              99.075223              99.109359
98.814548       98.805238    98.432845   98.370780      98.920060
```

[ ]: