ARTIFICIAL INTELLIGENCE

EMPLOYEE SALARY PREDICTION

Team members

Number	Name	ID	Department
1	Kamel Mahmoud Ahmed Nail	20171701076	BIO
2	Yasser Mohamed ElGhamry	20171701125	BIO
3	Mohamed Hesham Abdelaziz	20201701650	SWE
4	Nour Elden Mostafa	20201701660	SWE
5	Mohmed Alaa Baiomy	20201701649	SWE
6	Mohamed Ahmed Saied	20171701084	BIO

Preprocessing

If we look in the data, we will see a lot of ("?")

```
4       28            Private   338409  ...      40           Cuba   <=50K
...    ...                ...      ...  ...     ...            ...     ...
22787   55        Federal-gov    31965  ...      40  United-States   <=50K
22788   21            Private   143604  ...      29              ?   <=50K
22789   35            Private   174308  ...      40  United-States   <=50K
22790   31   Self-emp-not-inc   162551  ...      50              ?   <=50K
22791   39       Self-emp-inc   372525  ...      60  United-States    >50K
```

so, we replace it to NAN values using this line of code :

```
d.replace(" ?", np.nan, inplace=True)
```

For both files (Train , Test)

Preprocessing

The result is :

```
Run:   main

3         53           Private   234721  ...     40  United-States  <=50K
4         28           Private   338409  ...     40           Cuba  <=50K
...      ...               ...      ...  ...    ...            ...    ...
22787     55       Federal-gov    31965  ...     40  United-States  <=50K
22788     21           Private   143604  ...     29            NaN  <=50K
22789     35           Private   174308  ...     40  United-States  <=50K
22790     31  Self-emp-not-inc   162551  ...     50            NaN  <=50K
22791     39      Self-emp-inc   372525  ...     60  United-States   >50K
```

when we got NAN values , We can handle it with known techniques such as fillna() or dropna() method .

Preprocessing

We used both to trying to achieve the best score :

```
d = d.dropna(axis=0, how='any')
```

```
d.fillna(method='ffill', inplace=True)  # ffill backfill bfill pad
```

We change value on education from integers and string's to strings only to make best encoder on this column

```
# Convert "education" to string
d = d.astype({'education': 'string'})
```

Preprocessing

- prediction want column have the same name, so We change the column name in test data to the same in train data for prediction :

```
d.rename(columns={'workclass': 'work-class'}, inplace=True)
d.rename(columns={'fnlwgt': 'work-fnl'}, inplace=True)
d.rename(columns={'occupation': 'position'}, inplace=True)
```

Then we apply the labelencoder technique to convert the columns have strings to integer .

```
# ================================================================== #

FE_train_columns = ['work-class','education', 'marital-status','position', 'relationship','race','sex','native-country','salary']
FE_test_columns = ['work-class','education', 'marital-status', 'position','relationship','race','sex','native-country']

# ================================================================== #

label_encoder = preprocessing.LabelEncoder()
for i in FE_train_columns:
    final_train[i] = label_encoder.fit_transform(final_train[i])

test_label_encoder = preprocessing.LabelEncoder()
for c in FE_test_columns:
    final_test[c] = label_encoder.fit_transform(final_test[c])
```

Preprocessing

We searched for the best feature selection for categorical data and found that the mutual information technique is the best one for our data :

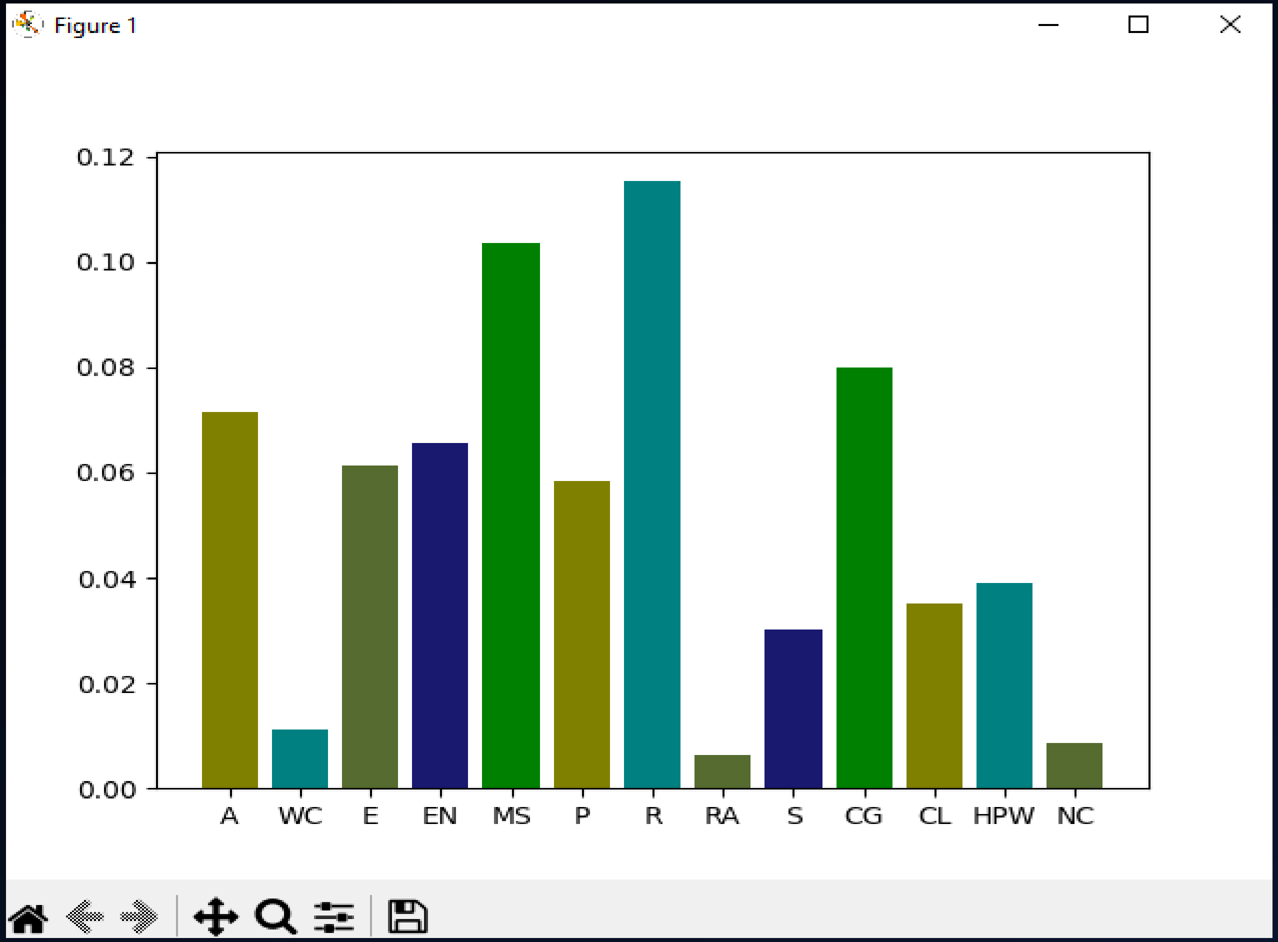
```
X=X.drop('salary', axis=1)
mutual_info = mutual_info_classif(X, Y)
mutual_info = pd.Series(mutual_info)
mutual_info.index = X.columns
mutual_info.sort_values(ascending=False)

# mutual_info_classif plot
listtt = list(mutual_info)
names = ['A','WC' , 'E' , 'EN','MS', 'P','R','RA','S','CG' ,'CL' ,'HPW','NC']
c = ['olive', 'teal', 'darkolivegreen', 'midnightblue', 'green']
plt.bar(names, listtt,color = c)
plt.show()

sel_five_cols = SelectKBest(mutual_info_classif , k = 12)#8 9 10 13 XGB
sel_five_cols.fit(X, Y)
train = X.columns[sel_five_cols.get_support()]
```

Preprocessing

The mutual information plotting :



Preprocessing

We use train test split() function to split the data to 20% test and 80% train data and store the result of this function in :

X_train

X_valid

Y_train

Y_valid

```
# ============================================================================ #

X_train, X_valid, y_train, y_valid = train_test_split(X, Y, random_state = 10 , test_size = 0.2, shuffle=True)

# =================================      (LR)      ========================================== #
```

The models

The best model achieved the best two accuracy on Kaggle was (XGBOOST) with the next csv files:

1) **XGB(The best).csv : 0.87235**

With parameters :

Dropna (train data)

Drop fnl-work column

Fillna('ffil') , (test data)

K=13 , Random State =10

The models

2) XGB(the second best).csv : 0.86996

With parameters :

Dropna (train data)

Drop fnl-work column

Fillna(‘ffil’) , (test data)

K=13 , Random State =8

Train & predict time for all algorithms