Salary Prediction Classification

A data science project by Rafif

About this project

In this project, I make machine learning models to predict whether a person makes > $50K or <= $50K a year

The dataset was obtained from Kaggle. It was acquired from the 1994 Census database by Barry Becker.

This project focuses on: data pre-processing, EDA, data visualization, feature selection, and classification algorithm.

Dataset

This dataset contains 15 columns:

- 6 numerical columns
- 8 categorical columns
- 1 target column (`salary`)

```
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   age             32561 non-null  int64 
 1   workclass       32561 non-null  object
 2   fnlwgt          32561 non-null  int64 
 3   education       32561 non-null  object
 4   education-num   32561 non-null  int64 
 5   marital-status  32561 non-null  object
 6   occupation      32561 non-null  object
 7   relationship    32561 non-null  object
 8   race            32561 non-null  object
 9   sex             32561 non-null  object
 10  capital-gain    32561 non-null  int64 
 11  capital-loss    32561 non-null  int64 
 12  hours-per-week  32561 non-null  int64 
 13  native-country  32561 non-null  object
 14  salary          32561 non-null  object
```

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	salary
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K

Pre-processing

Checking for any missing value (`NaN`)

```
1 # Check the number of NaN
2 # values in every column
3 df.isna().sum()
```

```
age               0
workclass         0
fnlwgt            0
education         0
education-num     0
marital-status    0
occupation        0
relationship      0
race              0
sex               0
capital-gain      0
capital-loss      0
hours-per-week    0
native-country    0
salary            0
```

No missing values? Sometimes they are marked as string characters (e.g. empty space ' ')!

```
Column: workclass

 Private             22696
 Self-emp-not-inc     2541
 Local-gov            2093
 ?                    1836
 State-gov            1298
 Self-emp-inc         1116
 Federal-gov           960
 Without-pay            14
 Never-worked            7
```

They are marked as '?'

Pre-processing

Replacing all ‘?’s with `NaN`s → there are 2399 (~7.4%) rows with missing values

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	salary
14	40	Private	121772	Assoc-voc	11	Married-civ-spouse	Craft-repair	Husband	Asian-Pac-Islander	Male	0	0	40	NaN	>50K
27	54	NaN	180211	Some-college	10	Married-civ-spouse	NaN	Husband	Asian-Pac-Islander	Male	0	0	60	South	>50K
38	31	Private	84154	Some-college	10	Married-civ-spouse	Sales	Husband	White	Male	0	0	38	NaN	>50K
51	18	Private	226956	HS-grad	9	Never-married	Other-service	Own-child	White	Female	0	0	30	NaN	<=50K
61	32	NaN	293936	7th-8th	4	Married-spouse-absent	NaN	Not-in-family	White	Male	0	0	40	NaN	<=50K

These rows are dropped altogether

Pre-processing

45 rows have duplicates

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	salary
2303	90	Private	52386	Some-college	10	Never-married	Other-service	Not-in-family	Asian-Pac-Islander	Male	0	0	35	United-States	<=50K
3917	19	Private	251579	Some-college	10	Never-married	Other-service	Own-child	White	Male	0	0	14	United-States	<=50K
4325	25	Private	308144	Bachelors	13	Never-married	Craft-repair	Not-in-family	White	Male	0	0	40	Mexico	<=50K
4767	21	Private	250051	Some-college	10	Never-married	Prof-specialty	Own-child	White	Female	0	0	10	United-States	<=50K
4881	25	Private	308144	Bachelors	13	Never-married	Craft-repair	Not-in-family	White	Male	0	0	40	Mexico	<=50K

Dropped the duplicates and kept the first row

Pre-processing

Binary encoding for `salary`

```
df['salary'] = df['salary'].apply(lambda x: 0 if x ==' <=50K' else 1)

df.head()
```

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	salary
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	0
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	0
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	0
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	0
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	0

Train-test split

To test the performance of my model-to-be with unseen data, I split the data into training and test sets with a ratio of 80:20.

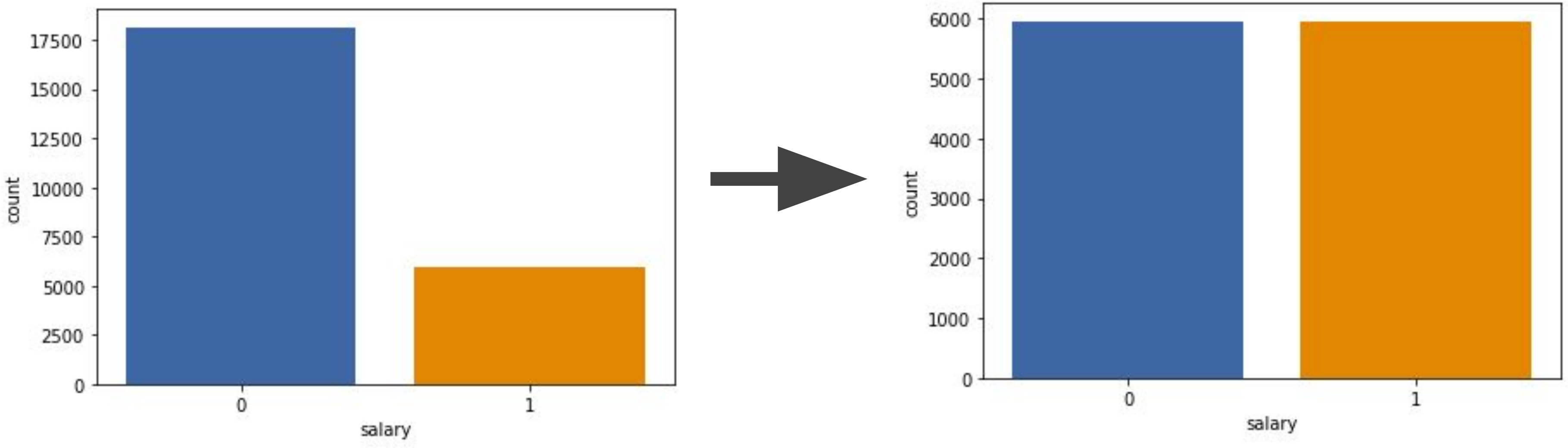
```
from sklearn.model_selection import train_test_split

# Select the features
X = df.drop('salary',axis=1)
# Select the target
y = df[['salary']]

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)
```

Data Understanding

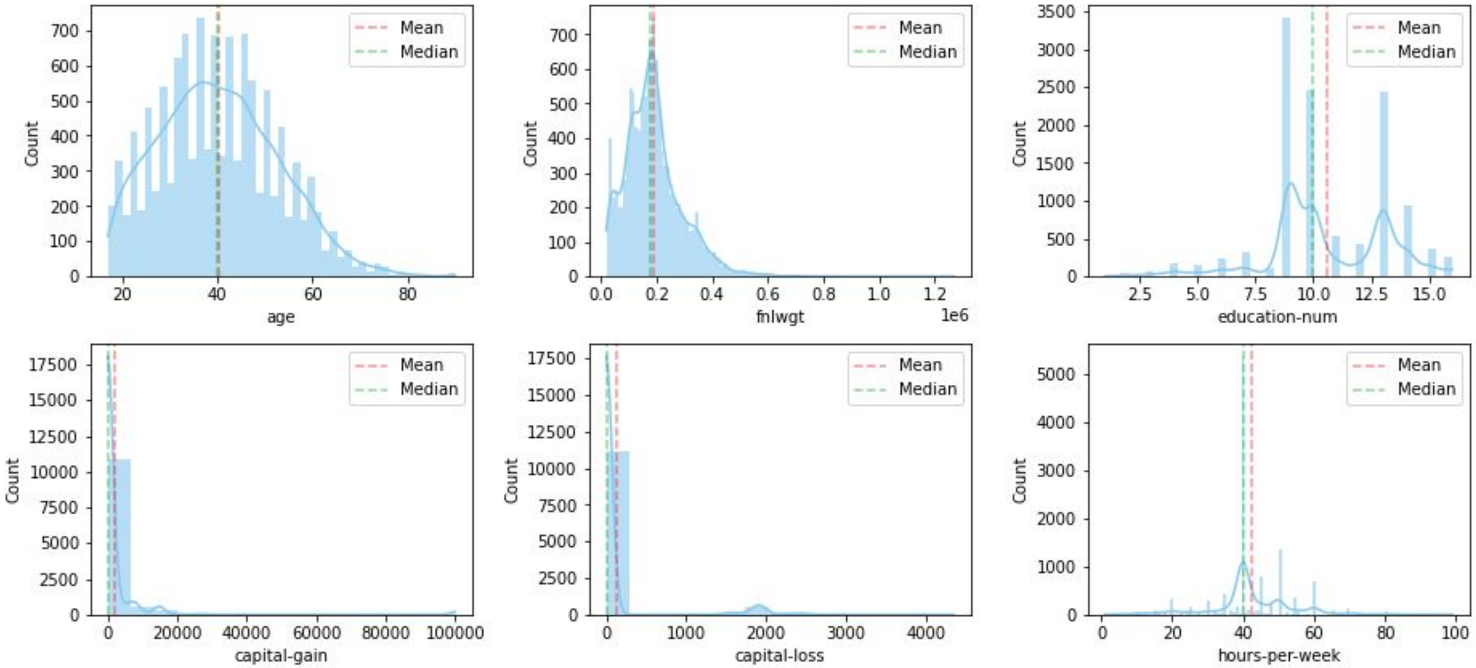
The target feature is not balanced (`salary=1` is ~3 times `salary=0`



Undersampling data with `salary` = 0

Data Understanding

Distribution of numerical features:

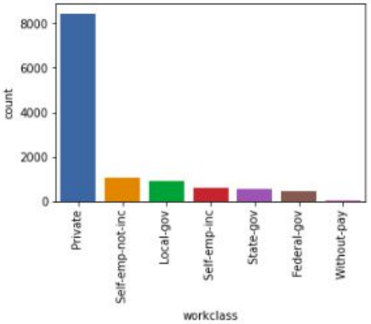


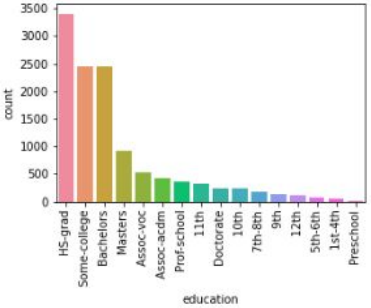
A lot of zeros in `capital-gain` and `capital-loss`

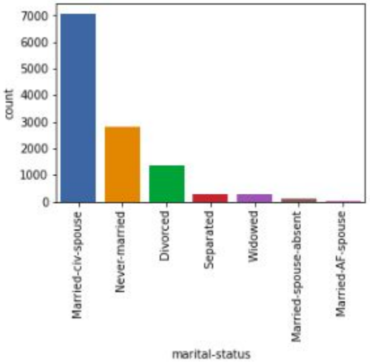
I kept them as they are

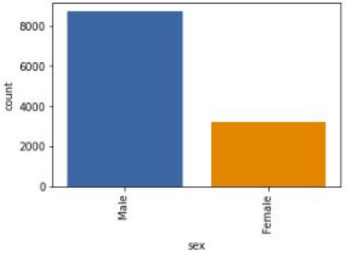
Data Understanding

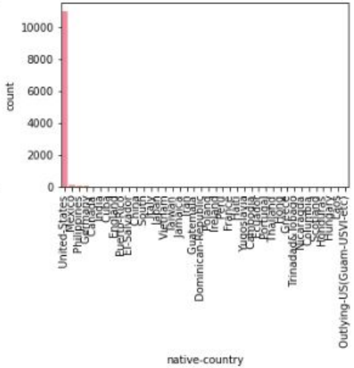
Count plot of categorical features:

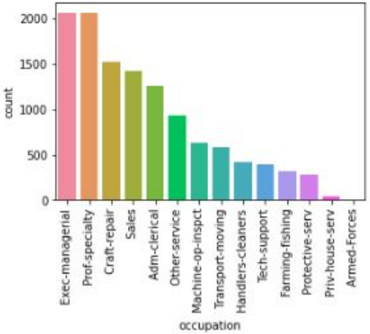


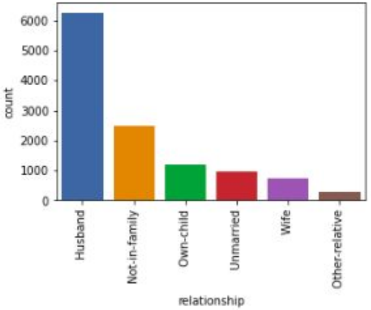


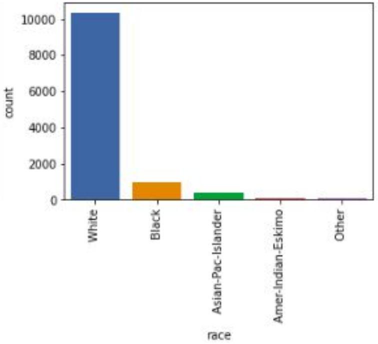






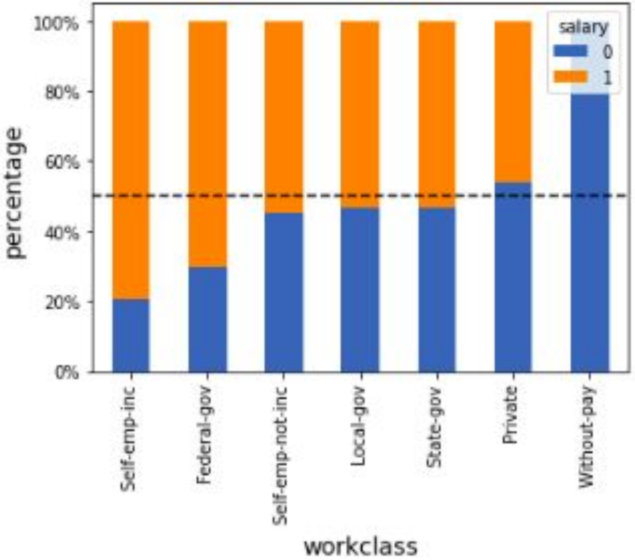


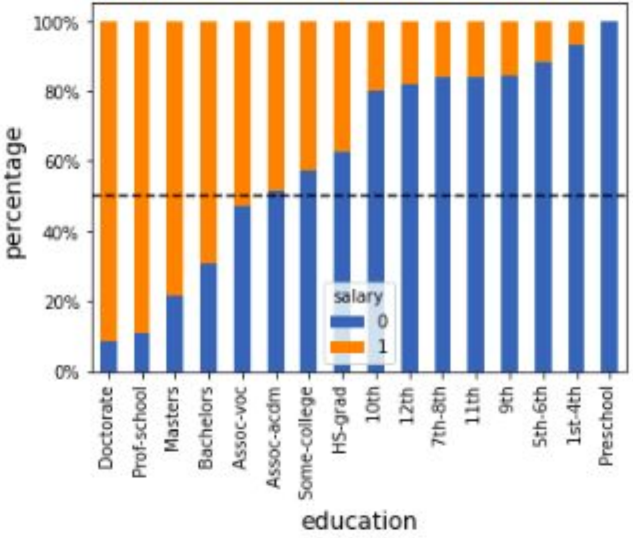




Data Understanding

The percentage of people based on their salary in each categorical value for `workclass` and `education`:





Self-employed people and doctoral graduates tend to make more salary

Overall there are correlations with `salary`

Data Understanding

No multicollinearity among numerical features

No clear distinction in all features based on `salary`.

Building the Models

- Not dropping any feature (the numerical features are weakly correlated with `salary` anyway)

- Classification algorithms: logistic regression and random forest

- Grid search with cross-validation to tune the parameters

Building the Models: logistic regression

Prerequisite I: label encoding using sklearn

```
# We first apply one hot encoding for all categorical features
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder

transformer = ColumnTransformer(transformers=[('cat', OneHotEncoder(drop='first',
                                                                    handle_unknown='ignore'),
                                                                    cat_col)]
                                              ,remainder='passthrough')

transformer.fit(X_train)
# Label encoding for training data
X_train_encoded = pd.DataFrame(transformer.transform(X_train).toarray(),columns=transformer.get_feature_names_out()
# Label encoding for test data
X_test_encoded = pd.DataFrame(transformer.transform(X_test).toarray(),columns=transformer.get_feature_names_out())
```

Prerequisite II: scale the features (to interpret the coefficients easier)

```
# To check feature importance later, we use the same scale across all features
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
scaler.fit(X_train_encoded)

X_train_scaled = scaler.transform(X_train_encoded)
X_test_scaled = scaler.transform(X_test_encoded)
```

Building the Models: logistic regression

```python
# Import the logistic regression model from sklearn
from sklearn.linear_model import LogisticRegression

# Import the grid search algorithm from sklearn
from sklearn.model_selection import GridSearchCV

import warnings
warnings.filterwarnings('ignore')

# parameter grid to be tested for the logistic regression model
parameters = {
    'penalty' : ['l1','l2'],
    'C'       : np.logspace(-2,2,5),
    'solver'  : ['newton-cg', 'lbfgs', 'liblinear'],
}

logreg = LogisticRegression()
clf = GridSearchCV(logreg,
                   param_grid=parameters,
                   scoring='accuracy',
                   cv=5)

clf.fit(X_train_scaled,y_train)
```

Grid search method to find the best parameters

Hyperparameters:

- Penalty
- C
- solver

```python
# Print the best model and its accuracy from the cross-validation method
print("Tuned Hyperparameters :", clf.best_params_)
print("Accuracy :",clf.best_score_)
```

```
Tuned Hyperparameters : {'C': 100.0, 'penalty': 'l1', 'solver': 'liblinear'}
Accuracy : 0.8192187227507215
```

Building the Models: logistic regression

	feature	feature_importance
15	cat__education_ Doctorate	4.492333
21	cat__marital-status_ Married-AF-spouse	4.106009
19	cat__education_ Prof-school	4.028426
17	cat__education_ Masters	3.526002
14	cat__education_ Bachelors	2.838709
22	cat__marital-status_ Married-civ-spouse	2.519159
12	cat__education_ Assoc-acdm	2.185667
13	cat__education_ Assoc-voc	2.003793
20	cat__education_ Some-college	1.617672
44	cat__relationship_ Wife	1.517337

Consistent with the results from EDA!

Building the Models: logistic regression

```
print(confusion_matrix(y_test,y_pred),'\n')
print(classification_report(y_test,y_pred))
```

```
[[4272  213]
 [ 901  642]]

              precision    recall  f1-score   support

           0       0.83      0.95      0.88      4485
           1       0.75      0.42      0.54      1543

    accuracy                           0.82      6028
   macro avg       0.79      0.68      0.71      6028
weighted avg       0.81      0.82      0.80      6028
```

```
print(f'accuracy = {accuracy_score(y_test,y_pred)}')
print(f'ROC AUC score = {roc_auc_score(y_test,y_pred)}')
```

```
accuracy = 0.8151957531519576
ROC AUC score = 0.6842904735378459
```

The accuracy is 81.5%, but the AUC score is only 68.4%

Since the test `salary` is also very likely to be imbalanced, this indicates that this model is not good enough

Building the Models: random forest

```
# Import random forest from sklearn
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(random_state=7)
parameters = {'n_estimators':[100,200,300,400,500],
              'criterion': ['gini','entropy']}

clf = GridSearchCV(estimator=rf,
                   param_grid=parameters,
                   scoring='accuracy',
                   cv=5)

clf.fit(X_train_encoded,y_train)
```

Grid search method to find the best parameters

Hyperparameters:

- n_estimators
- criterion

```
# Print the best model and its accuracy from the cross-validation method
print("Tuned Hyperparameters :", clf.best_params_)
print("Accuracy :",clf.best_score_)
```

```
Tuned Hyperparameters : {'criterion': 'entropy', 'n_estimators': 200}
Accuracy : 0.8198900996553972
```

Building the Models: random forest

```
print(confusion_matrix(y_test,y_predrf),'\n')
print(classification_report(y_predrf, y_test))
```

```
[[3605  880]
 [ 243 1300]]

              precision    recall  f1-score   support

           0       0.80      0.94      0.87      3848
           1       0.84      0.60      0.70      2180

    accuracy                           0.81      6028
   macro avg       0.82      0.77      0.78      6028
weighted avg       0.82      0.81      0.80      6028
```

```
print(f'accuracy = {accuracy_score(y_test,y_predrf)}')
print(f'ROC AUC score = {roc_auc_score(y_test,y_predrf)}')
```

```
accuracy = 0.8137027206370272
ROC AUC score = 0.8231524972346072
```

The accuracy and AUC score are both >80%, indicating this model performs quite well.

This is expected since the random forest algorithm is more complex than logistic regression

Conclusions

1. This dataset contains 15 columns, 6 are numerical, 8 are categorical, and the target `salary` is binary.
2. The numerical features are very weakly correlated with `salary`, and the categorical features seem to have stronger correlations with `salary`.
3. I use two classification algorithms: logistic regression and random forest to predict `salary`.
4. The best model is the random forest classifier, where I can obtain an accuracy of 81.4%, and an ROC AUC score of 82.3%.