FINDING THE RIGHT CUSTOMER

Presenter：Rita, Terrell and Lionel

CONTENTS

Aim and Ideal of the Model

01

- Create a model to find whether the customers can afford our product
- Majority of the old customers have a yearly income over $50,000
- This model helps us to predict whether the new customers make $50,000 a year, so that we can decide if we should promote the product to them

Dataset

The dataset contains:

- Age
- Work-class
- Education
- Martial-status
- Occupation
- Relationship
- Race
- Sex
- Capital-gain
- Capital-loss
- Work hours per week
- Native-country
- Income

Dataset Cleaning

- 4265 Missing values
- Drop rows that contain missing values
- Since we still have a large sample size of 30126

```
df = df.replace(" ?",np.nan)
```

```
df.isna().sum()
```

```
age                  0
workclass         1836
fnlwgt               0
education            0
education-num        0
marital-status       0
occupation        1843
relationship         0
race                 0
sex                  0
capital-gain         0
capital-loss         0
hours-per-week       0
native-country     583
income               0
income2              0
dtype: int64
```

Method

Data Exploration and Visualization

Feature Engineering

Create Model

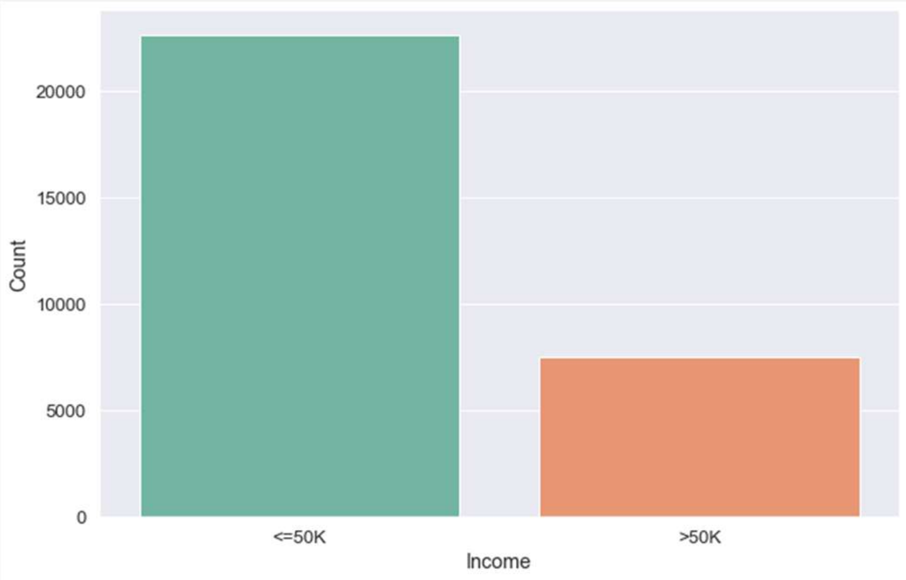
Resampling by SMOTEENN

SVM / Xgboost

Conclusion

Data Exploration and Visualization

How much do they earn?

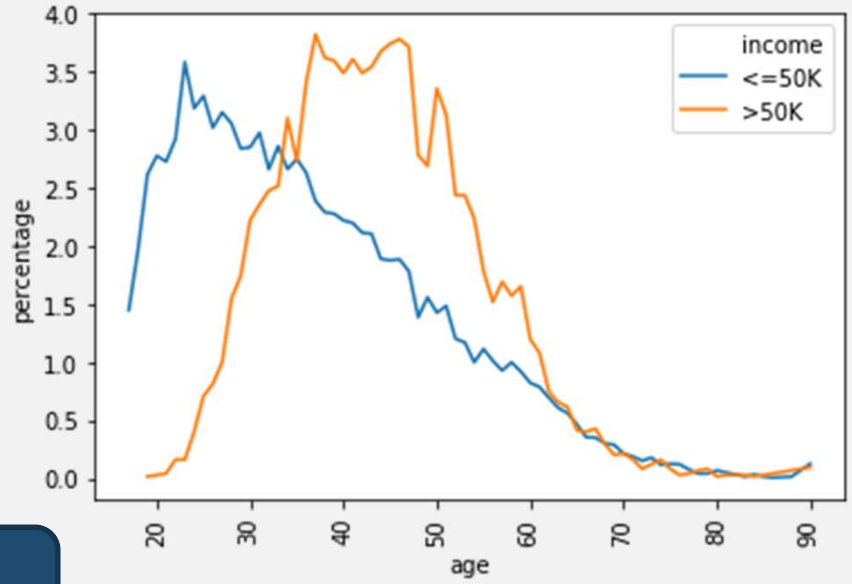


- 75.1% of them have an income of less or equal to 50K
- 24.9% of them have an income of greater than 50K

Data Exploration and Visualization

Age Group

- Majority of the population in this sample is in the age of 20-50
- Age 20-33: more people earn less than 50K
- Age 33-60: more people earn more than 50K
- Age 60+: about the same

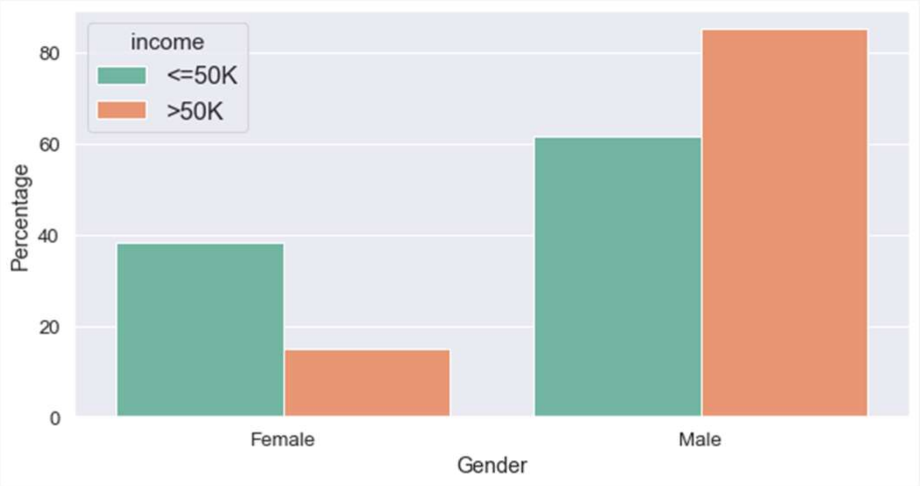


Prediction 1: Age is an important feature

Data Exploration and Visualization

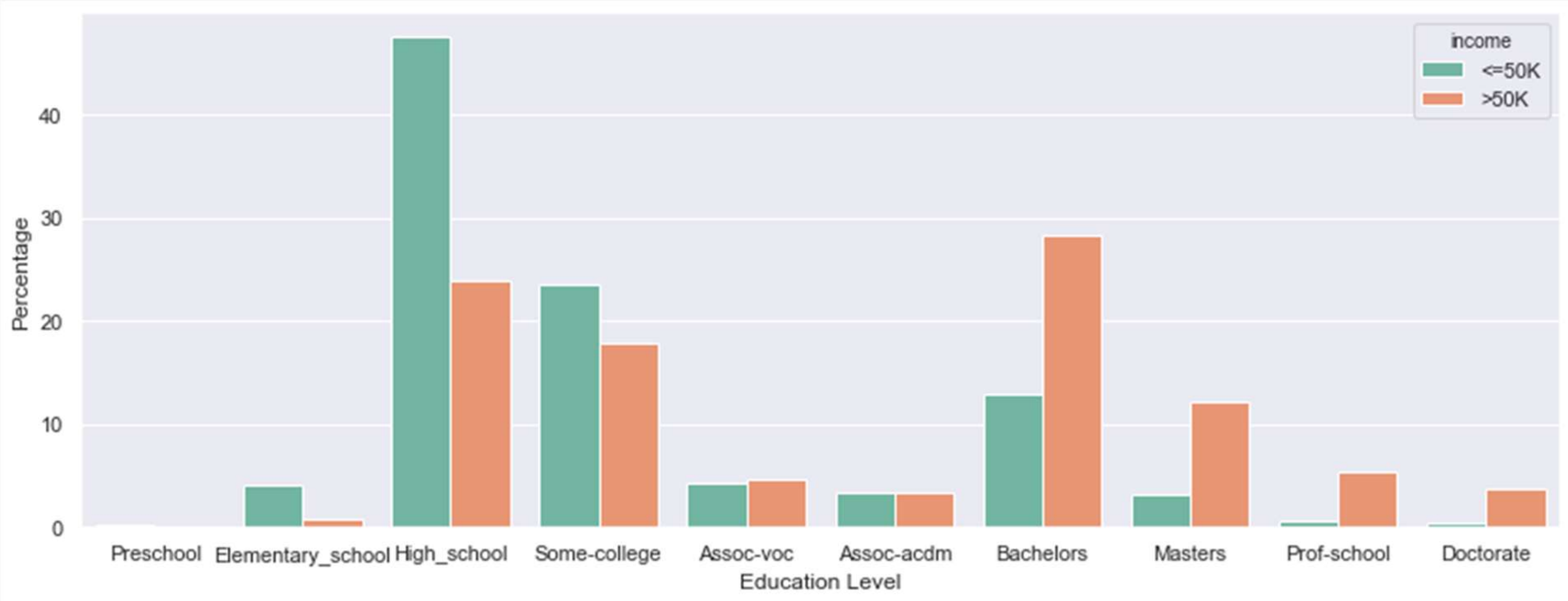
Male or Female

- More than 80% of the people who are making more than $50,000 a year are male



Prediction: More male customers than female

Data Exploration and Visualization



Holder of a Bachelor's Degree or higher seems to earn more

STUDY HARD!

Feature Engineering

```
df.workclass.value_counts()
```

```
Private             22286
Self-emp-not-inc     2499
Local-gov            2067
State-gov            1279
Self-emp-inc         1074
Federal-gov           943
Without-pay            14
Name: workclass, dtype: int64
```

```
self_employed = ["Self-emp-not-inc","Self-emp-inc"]
gov = ["Local-gov","State-gov", "Federal-gov"]

df["workclass"].replace(to_replace = self_employed, value = "self-employed", inplace = True)
df["workclass"].replace(to_replace = gov, value = "gov", inplace = True)

df["workclass"].value_counts()
```

```
Private          22286
gov               4289
self-employed     3573
Without-pay         14
Name: workclass, dtype: int64
```

We have also done feature engineering on the following features:

Education, Native-country ,Martial Status & Age

Feature Scaling

```
dum_df = pd.get_dummies(df, prefix_sep='_', drop_first=True)
dum_df.head()
```

	fnlwgt	capital-gain	capital-loss	hours-per-week	education_num	age_30-39	age_40-49	age_50-59	age_<20	age_>60	...	relationship_Own-child	relationship_Unmarried	relationship_V
0	77516	2174	0	40	7	1	0	0	0	0	...	0	0	
1	83311	0	0	13	7	0	0	1	0	0	...	0	0	
2	215646	0	0	40	3	1	0	0	0	0	...	0	0	
3	234721	0	0	40	3	0	0	1	0	0	...	0	0	
4	338409	0	0	40	7	0	0	0	0	0	...	0	0	

5 rows × 41 columns

```
X_train = dum_df.drop(columns = "income_>50K")
y_train = dum_df["income_>50K"]
```

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 1))
rescaledX = scaler.fit_transform(X_train)
```

Create Model

Logistic Regression

- Predict ≤ $50,000 ,very well
- Predict > $50,000 , OK but not good enough
- We would have lost 1492 potential customers under this model
- Imbalance dataset
- Focus on the recall

```
print(classification_report(y_train,y_pred_train))

              precision    recall  f1-score   support

           0       0.87      0.93      0.90     22654
           1       0.74      0.60      0.66      7508

    accuracy                           0.85     30162
   macro avg       0.81      0.76      0.78     30162
weighted avg       0.84      0.85      0.84     30162
```

```
print(classification_report(y_test,y_pred_test))

              precision    recall  f1-score   support

           0       0.88      0.93      0.90     11360
           1       0.73      0.60      0.66      3700

    accuracy                           0.85     15060
   macro avg       0.80      0.76      0.78     15060
weighted avg       0.84      0.85      0.84     15060
```

```
from sklearn.metrics import confusion_matrix
tn, fp, fn, tp = confusion_matrix(y_test, y_pred_test).ravel()
print([tp,fp])
print([fn,tn])

[2208, 816]
[1492, 10544]
```

		Actual Class	
		>50k	<=50k
Predicted Class	>50k	2208 True Positives	816 False Positives
	<=50k	1492 False Negatives	10544 True Negatives

Resampling

Before Resampling	SMOTEENN	SMOTE

Before Resampling

```
print(classification_report(y_train,y_pred_train))
              precision    recall  f1-score   support

           0       0.87      0.93      0.90     22654
           1       0.74      0.60      0.66      7508

    accuracy                           0.85     30162
   macro avg       0.81      0.76      0.78     30162
weighted avg       0.84      0.85      0.84     30162
```

```
print(classification_report(y_test,y_pred_test))
              precision    recall  f1-score   support

           0       0.88      0.93      0.90     11360
           1       0.73      0.60      0.66      3700

    accuracy                           0.85     15060
   macro avg       0.80      0.76      0.78     15060
weighted avg       0.84      0.85      0.84     15060
```

0.73

SMOTEENN

```
print(classification_report(y_resampled,y_pred_train_re))
              precision    recall  f1-score   support

           0       0.94      0.93      0.93     15678
           1       0.93      0.94      0.93     15599

    accuracy                           0.93     31277
   macro avg       0.93      0.93      0.93     31277
weighted avg       0.93      0.93      0.93     31277
```

```
print(classification_report(y_test,y_pred_test_re))
              precision    recall  f1-score   support

           0       0.95      0.76      0.84     11360
           1       0.54      0.87      0.66      3700

    accuracy                           0.78     15060
   macro avg       0.74      0.81      0.75     15060
weighted avg       0.85      0.78      0.80     15060
```

0.87

SMOTE

```
In [63]: print(classification_report(y_smote,y_pred_train_re2))
              precision    recall  f1-score   support

           0       0.85      0.79      0.82     22654
           1       0.80      0.86      0.83     22654

    accuracy                           0.82     45308
   macro avg       0.83      0.82      0.82     45308
weighted avg       0.83      0.82      0.82     45308
```

```
In [64]: print(classification_report(y_test,y_pred_test_re2))
              precision    recall  f1-score   support

           0       0.94      0.79      0.86     11360
           1       0.57      0.84      0.68      3700

    accuracy                           0.80     15060
   macro avg       0.75      0.82      0.77     15060
weighted avg       0.85      0.80      0.81     15060
```

0.84

Precision ✓	Recall ✓	Recall ✓
Accuracy ✓		

KNN – SVM

Logistic regression

	precision	recall	f1-score	support
0	0.95	0.76	0.84	11360
1	0.54	0.87	0.66	3700
accuracy			0.78	15060
macro avg	0.74	0.81	0.75	15060
weighted avg	0.85	0.78	0.80	15060

KNN

	precision	recall	f1-score	support
0	0.91	0.79	0.84	11360
1	0.54	0.75	0.63	3700
accuracy			0.78	15060
macro avg	0.72	0.77	0.74	15060
weighted avg	0.82	0.78	0.79	15060

SVM

	precision	recall	f1-score	support
0	0.91	0.71	0.80	11360
1	0.47	0.77	0.58	3700
accuracy			0.73	15060
macro avg	0.69	0.74	0.69	15060
weighted avg	0.80	0.73	0.75	15060

Combine over- and under-sampling using SMOTEENN

XGboost

	precision	recall	f1-score	support
0	0.94	0.83	0.88	11360
1	0.62	0.83	0.71	3700
accuracy			0.83	15060
macro avg	0.78	0.83	0.79	15060
weighted avg	0.86	0.83	0.84	15060

- Best result
- Relatively high precision
- High recall and accuracy

Tuning

```
from sklearn.model_selection import GridSearchCV
grid_search=GridSearchCV(estimator=xgbc, param_grid=dict(learning_rate=[0.01,0.1],max_depth=[3,7]),scoring="recall").fit(rescalec
print("tuned hpyerparameters :(best parameters) ",grid_search.best_params_)
print("accuracy :",grid_search.best_score_)
```

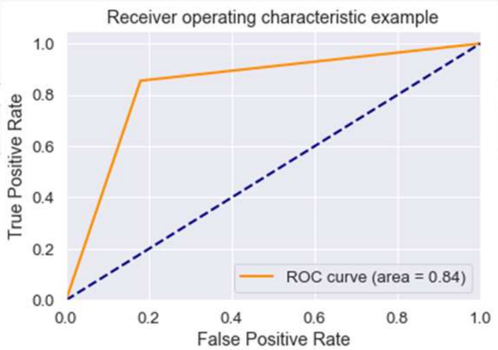
```
tuned hpyerparameters :(best parameters)  {'learning_rate': 0.01, 'max_depth': 3}
accuracy : 0.8560212410545655
```

We found that 'learning_rate'=0.01 and 'max_depth'=3 are the best parameters

XGBoost

```
xgbc = XGBClassifier(n_estimators=5000,n_jobs=-1, learning_rate= 0.01,max_depth= 3, random_state=2,scale_pos_weight=3)

xgbc.fit(rescaledX,y_train)
y_pred_xgbc_test = xgbc.predict(rescaledX_test)
y_pred_xgbc_train = xgbc.predict(rescaledX)
```




```
print(classification_report(y_train,y_pred_xgbc_train))
```

	precision	recall	f1-score	support
0	0.95	0.83	0.89	22654
1	0.63	0.88	0.73	7508
accuracy			0.84	30162
macro avg	0.79	0.85	0.81	30162
weighted avg	0.87	0.84	0.85	30162

```
print(classification_report(y_test,y_pred_xgbc_test))
```

	precision	recall	f1-score	support
0	0.95	0.82	0.88	11360
1	0.61	0.86	0.71	3700
accuracy			0.83	15060
macro avg	0.78	0.84	0.79	15060
weighted avg	0.86	0.83	0.84	15060

Before tuning

	precision	recall	f1-score	support
0	0.94	0.83	0.88	11360
1	0.62	0.83	0.71	3700
accuracy			0.83	15060
macro avg	0.78	0.83	0.79	15060
weighted avg	0.86	0.83	0.84	15060

XGBoost

Before

		Actual Class	
		>50k	<=50k
Predicted Class	>50k	2208 True Positives	816 False Positives
	<=50k	1492 False Negatives	10544 True Negatives

After

		Actual Class	
		>50k	<=50k
Predicted Class	>50k	3164 True Positives	2040 False Positives
	<=50k	536 False Negatives	9320 True Negatives

After tuning with XGBoost:

1. We have predicted 956 target customers more than before
2. Loss of potential customers goes down from 1492 to 536

Feature Importance

Improvement

- Some features have a strong correlation between them, such as race and native-country, relationship and marital-status

In this model, we put them as 2 separate features which might enhance / reduce the importance of it

- We would have removed features with the lowest importance

THANK YOU!

Presenter：Rita, Terrell and Lionel