- **<u>Processing the cleaned data</u>**

Before training our dataset for the best result we need to process it accordingly

Scaling -

min max scaling

```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
numerical=['age','education_num','capital_gain','capital_loss','hours_per_week']
data[numerical] = scaler.fit_transform(data[numerical])
data1=data
data
```

	id	age	workclass	education	education_num	marital_status	occupation	relationship	race	sex	capital_gain	capital_loss	hours_per_we
0	12106	0.226667	Private	HS-grad	0.533333	Divorced	Adm-clerical	Other-relative	White	Female	0.500000	0.360634	0.1566

Changing categorical to numerical for training data

```
#Changing categorical to ordinal
data['over_50k'] = data['over_50k'].map({'<=50K': 0, '>50K': 1}).astype(int)
data['sex'] = data['sex'].map({'Male': 0, 'Female': 1}).astype(int)
data['race'] = data['race'].map({'Black': 0, 'Asian-Pac-Islander': 1,'Other': 2, 'White': 3, 'Amer-Indian-Eskimo': 4}).astyp
data['marital_status'] = data['marital_status'].map({'Married-spouse-absent': 0, 'Widowed': 1, 'Married-civ-spouse': 2, 'Sep
data['workclass'] = data['workclass'].map({'Self-emp-inc': 0, 'State-gov': 1,'Federal-gov': 2, 'Without-pay': 3, 'Local-gov'
data['education'] = data['education'].map({'Some-college': 0, 'Preschool': 1, '5th-6th': 2, 'HS-grad': 3, 'Masters': 4, '12t
data['relationship'] = data['relationship'].map({'Not-in-family': 0, 'Wife': 1, 'Other-relative': 2, 'Unmarried': 3,'Husband
data['occupation'] = data['occupation'].map(
    { 'Farming-fishing': 1, 'Tech-support': 2, 'Adm-clerical': 3, 'Handlers-cleaners': 4,
 'Prof-specialty': 5,'Machine-op-inspct': 6, 'Exec-managerial': 7,'Priv-house-serv': 8,'Craft-repair': 9,'Sales': 10, 'Trans

data.head()

```

Dropping some unwanted columns

From 15 to 14 columns

Removed id column

Removing outliner

Using Z score –

1971 outliners are dropped

```
z = np.abs(stats.zscore(data))
dataz = data[(z < 3).all(axis=1)]
print(data.shape,dataz.shape)
sns.boxplot(data=dataz)

```

```
(22084, 13) (20113, 13)
```

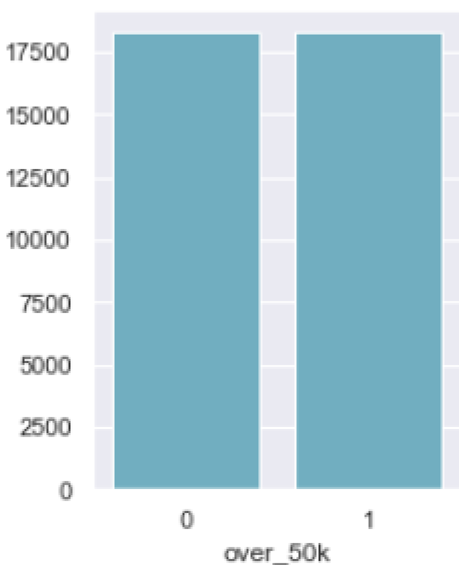
Solving Data Imbalance

Tried few techniques of upsampling and downlsampling .Best result came from SMOTE

using SMOTE to create data balance

```
from imblearn.over_sampling import SMOTE

smote = SMOTE()
X_sm, y_sm = smote.fit_sample(X, y)
sns.countplot(x="over_50k", data=pd.DataFrame(y_sm), color="c");
```



- **<u>Dividing data in test and train dataset for analysis</u>**

  we have tried train test split and kfold. Kfold give better results so we use it

Using K Fold

```python
from sklearn.model_selection import KFold
kf=KFold(n_splits=10, random_state=42, shuffle=False)

# X is the feature set and y is the target
for train_index, test_index in kf.split(X,y):
   #print("Train:", train_index, "Validation:", val_index)
    X_train, X_test = X.iloc[train_index,:], X.iloc[test_index,:]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

```

- <u>**Training and model selection**</u>

Evaluation parameter

we have created a function to give all evaluation parameters by one call

```python
evaluation=pd.DataFrame()
```

```python

def print_scores(y_test,y_pred,y_pred_prob):

    print('test-set confusion matrix:\n', confusion_matrix(y_test,y_pred))
    print("recall score: ", recall_score(y_test,y_pred))
    print("precision score: ", precision_score(y_test,y_pred))
    print("f1 score: ", f1_score(y_test,y_pred))
    print("accuracy score: ", accuracy_score(y_test,y_pred))
    print("ROC AUC: {}".format(roc_auc_score(y_test, y_pred_prob)))

    # Compute predicted probabilities: y_pred_prob


    # Generate ROC curve values: fpr, tpr, thresholds
    fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)

    # Plot ROC curve
    import matplotlib.pyplot as plt
    plt.plot([0, 1], [0, 1], 'k--')
    plt.plot(fpr, tpr)
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('ROC Curve')
    plt.show()
    return [recall_score(y_test,y_pred),precision_score(y_test,y_pred),f1_score(y_test,y_pred),accuracy_score(y_test,y_pred)
```

Prediction function

We have created a function to predict any model

```python

def get_predictions(clf, X_train, y_train, X_test):
    # create classifier
    clf = clf
    # fit it to training data
    clf.fit(X_train,y_train)
    # predict using test data
    y_pred = clf.predict(X_test)
    # Compute predicted probabilities: y_pred_prob
    y_pred_prob = clf.predict_proba(X_test)[:,1]
    #y_pred_prob = clf.predict_proba(X_test)
    # train-set predictions
    train_pred = clf.predict(X_train)
    print('train-set confusion matrix:\n', confusion_matrix(y_train,train_pred))
    return y_pred, y_pred_prob
```

Analyzing different models

we have used naïve bais,logistic regression,random forest,xgboost, and mlp(Artificial neural network)

	recall_score	precision_score	f1_score	accuracy_score	roc_auc_score	model
0	0.669269	0.820793	0.737327	0.766202	0.872270	GaussianNB
1	0.854992	0.719718	0.781545	0.765655	0.833932	LogisticRegression
2	0.909649	0.887378	0.898375	0.899098	0.962921	Randomforest
3	0.923592	0.907895	0.915676	0.916598	0.980339	xgboost
4	0.865588	0.824217	0.844396	0.843588	0.917406	ANN

After the analysis best result came from xgboost

Fine tuning

We try to fine tune the xgboost model at different values using gridsearchcv at 15 other models of xgboost but best result are coming at default value itself

We are getting max 98 % accuracy and around 85-95 accuracy by every model