STROKE PREDICTION

CHENG SIN YI

CONTENT

1. Introduction to dataset
2. Feature engineering
3. EDA
4. Model training
5. Feature importance
6. Hyperparameter tuning – GridSearchCV & RandomSearchCV

Dataset

	A	B	C	D	E	F	G	H	I	J	K
1	**gender**	**age**	**hypertension**	**heart_disease**	**ever_married**	**work_type**	**Residence_type**	**avg_glucose_level**	**bmi**	**smoking_status**	**stroke**
2	Male	67	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
3	Male	80	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
4	Female	49	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
5	Female	79	1	0	Yes	Self-employed	Rural	174.12	24	never smoked	1
6	Male	81	0	0	Yes	Private	Urban	186.21	29	formerly smoked	1
7	Male	74	1	1	Yes	Private	Rural	70.09	27.4	never smoked	1
8	Female	69	0	0	No	Private	Urban	94.39	22.8	never smoked	1
9	Female	78	0	0	Yes	Private	Urban	58.57	24.2	Unknown	1
10	Female	81	1	0	Yes	Private	Rural	80.43	29.7	never smoked	1
11	Female	61	0	1	Yes	Govt_job	Rural	120.46	36.8	smokes	1
12	Female	54	0	0	Yes	Private	Urban	104.51	27.3	smokes	1
13	Female	79	0	1	Yes	Private	Urban	214.09	28.2	never smoked	1

- 5110 rows. 209 rows with stroke – imbalanced

PROBLEM STATEMENT

Stroke has already reached epidemic proportions. Globally 1 in 4 adults over the age of 25 will have a stroke in their lifetime. There are many other risk factors, including tobacco use, physical inactivity, unhealthy diet and harmful use of alcohol which have been found to increase the risk of stroke. The incidence of stroke has also been found to increase significantly with age. This dataset is used to predict whether a patient is likely to get stroke based on the input parameters like gender, age, various diseases, and smoking status, as well as which ML model would be the best suited to be used for the prediction.

DATA CLEANING

- Removing rows where BMI values are missing
- 5110 -> 4909

```
df.dropna(inplace = True)
df.drop('id',axis = 1, inplace = True)
df
```

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
2	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
3	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
4	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1
5	Male	81.0	0	0	Yes	Private	Urban	186.21	29.0	formerly smoked	1
...	...	...	...	...	...	...	...	...	...	...	...

DATA CLEANING

- For simplicity's sake, replace 'other' with the mode.

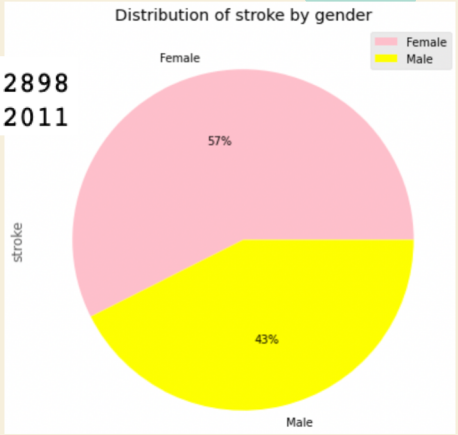
```
df['gender'].value_counts()
```

```
Female    2897
Male      2011
Other        1
Name: gender, dtype: int64
```

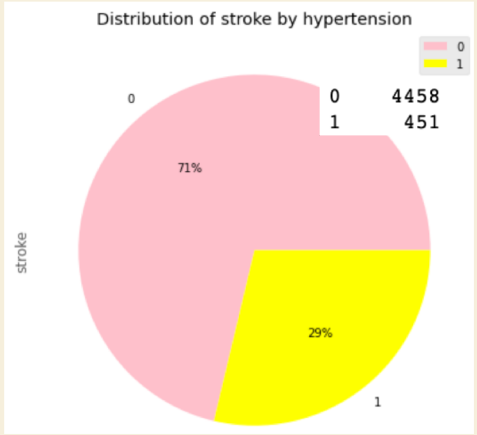
```
df['gender'].replace('Other',df['gender'].mode().values[0], inplace = True)
```

```
df['gender'].value_counts()
```

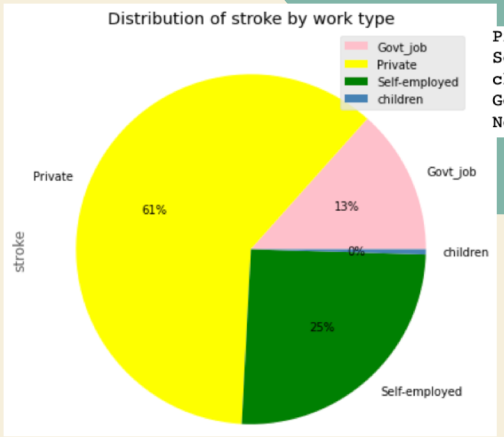
```
Female    2898
Male      2011
Name: gender, dtype: int64
```



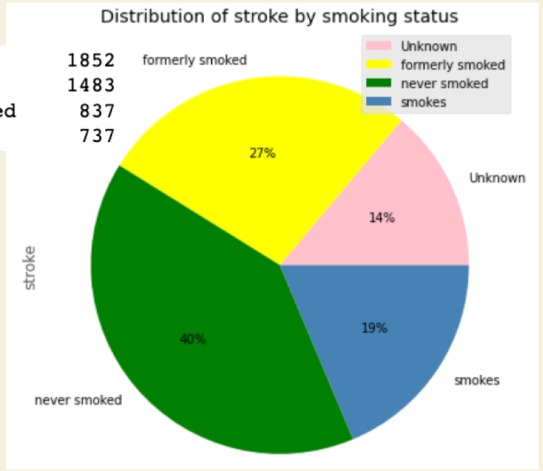
Female	2898
Male	2011



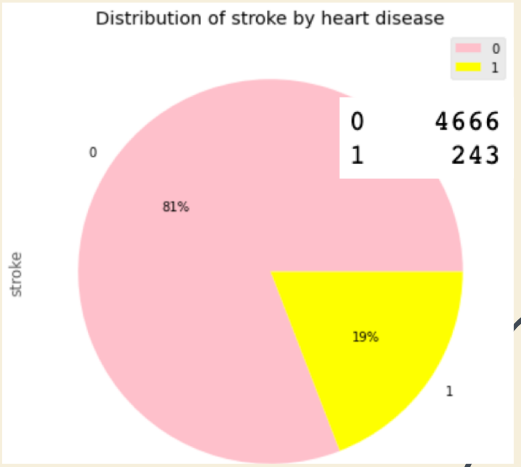
0	4458
1	451



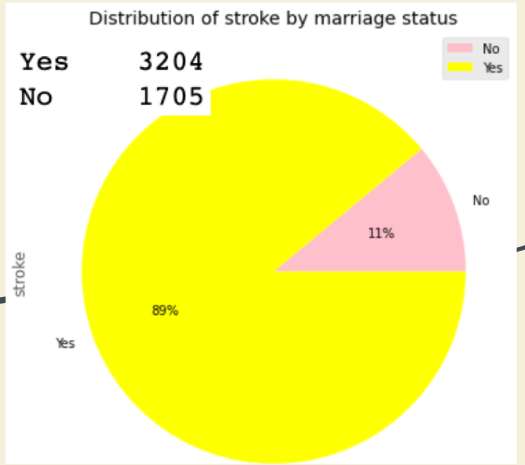
Private	2811
Self-employed	775
children	671
Govt_job	630
Never_worked	22



never smoked	1852
Unknown	1483
formerly smoked	837
smokes	737

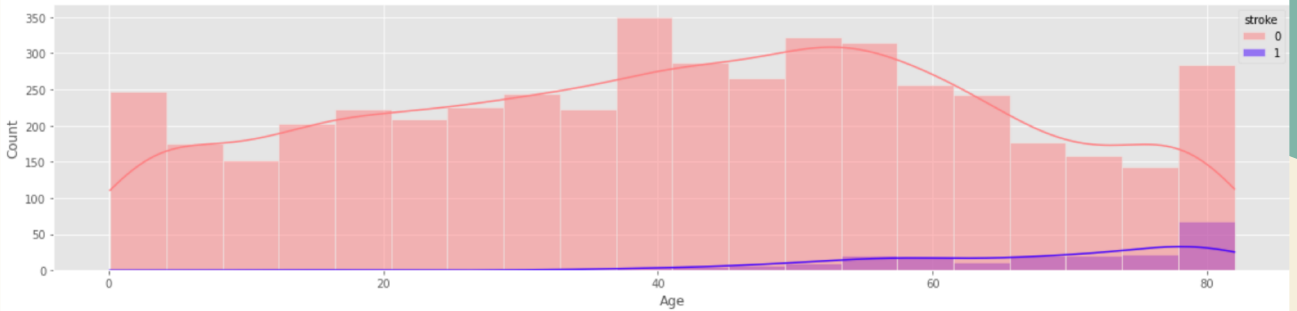


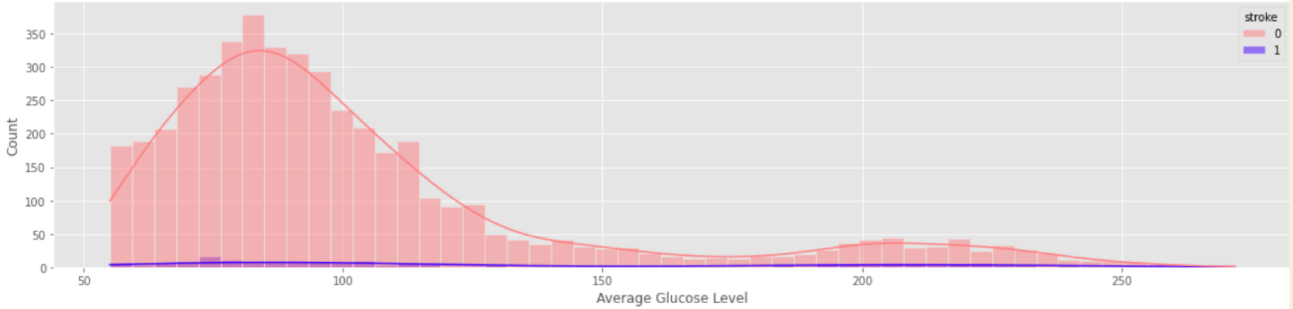
0	4666
1	243

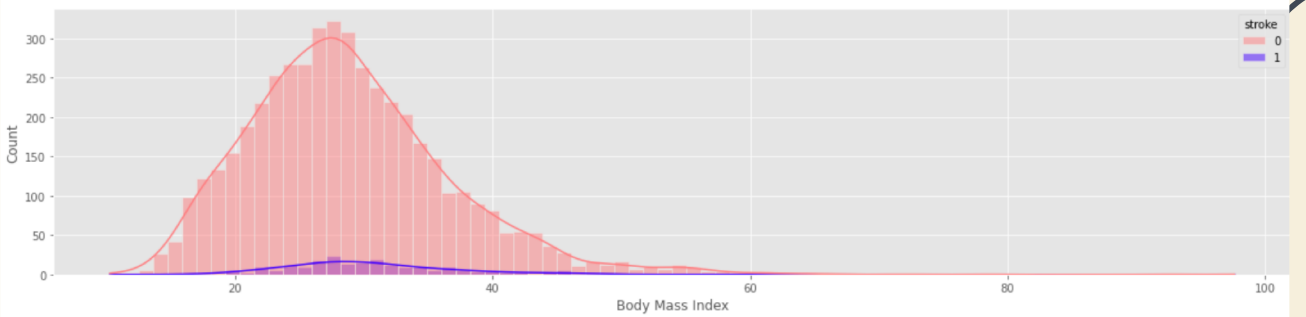


Yes	3204
No	1705

Distribution of numerical columns on stroke







Logit Regression Results

Dep. Variable:	stroke	No. Observations:	4909
Model:	Logit	Df Residuals:	4894
Method:	MLE	Df Model:	14
Date:	Sat, 17 Sep 2022	Pseudo R-squ.:	0.08540
Time:	01:05:33	Log-Likelihood:	-790.39
converged:	True	LL-Null:	-864.19
Covariance Type:	nonrobust	LLR p-value:	2.166e-24

	coef	std err	z	P>|z|	[0.025	0.975]
age	0.0324	0.004	7.246	0.000	0.024	0.041
hypertension	0.9885	0.178	5.566	0.000	0.640	1.337
heart_disease	0.7460	0.212	3.517	0.000	0.330	1.162
avg_glucose_level	0.0032	0.001	2.328	0.020	0.001	0.006
bmi	-0.1275	0.010	-12.976	0.000	-0.147	-0.108
gender_Male	-0.1353	0.148	-0.915	0.360	-0.425	0.155
ever_married_Yes	-0.5080	0.192	-2.641	0.008	-0.885	-0.131
work_type_Never_worked	-152.8843	1.25e+33	-1.22e-31	1.000	-2.45e+33	2.45e+33
work_type_Private	-0.7651	0.165	-4.639	0.000	-1.088	-0.442
work_type_Self-employed	-0.8595	0.213	-4.039	0.000	-1.277	-0.442
work_type_children	-4.3629	1.015	-4.299	0.000	-6.352	-2.374
Residence_type_Urban	-0.2747	0.139	-1.970	0.049	-0.548	-0.001
smoking_status_formerly smoked	-0.2614	0.207	-1.265	0.206	-0.666	0.144
smoking_status_never smoked	-0.5674	0.182	-3.115	0.002	-0.924	-0.210
smoking_status_smokes	-0.2958	0.222	-1.332	0.183	-0.731	0.139

	OR	Lower CI	Upper CI
age	1.032954e+00	1.023934	1.042052
hypertension	2.687269e+00	1.897358	3.806037
heart_disease	2.108461e+00	1.391396	3.195070
avg_glucose_level	1.003226e+00	1.000509	1.005950
bmi	8.803301e-01	0.863545	0.897442
gender_Male	8.734787e-01	0.653650	1.167238
ever_married_Yes	6.017253e-01	0.412722	0.877282
work_type_Never_worked	4.010608e-67	0.000000	inf
work_type_Private	4.652951e-01	0.336789	0.642835
work_type_Self-employed	4.233642e-01	0.278976	0.642482
work_type_children	1.274127e-02	0.001743	0.093135
Residence_type_Urban	7.598031e-01	0.578126	0.998573
smoking_status_formerly smoked	7.699771e-01	0.513573	1.154393
smoking_status_never smoked	5.669792e-01	0.396730	0.810287
smoking_status_smokes	7.439550e-01	0.481488	1.149498

Previous conclusion

- Age, hypertension, heart disease, average glucose level and BMI are found to be statistically significant.

Scale features columns

```
scaler = StandardScaler()
X = scaler.fit_transform(X)
```

Random Over Sampler

```
np.unique(Y_train, return_counts=True)
```

```
(array([0, 1]), array([2831, 2831]))
```

Splitting into test and train dataset

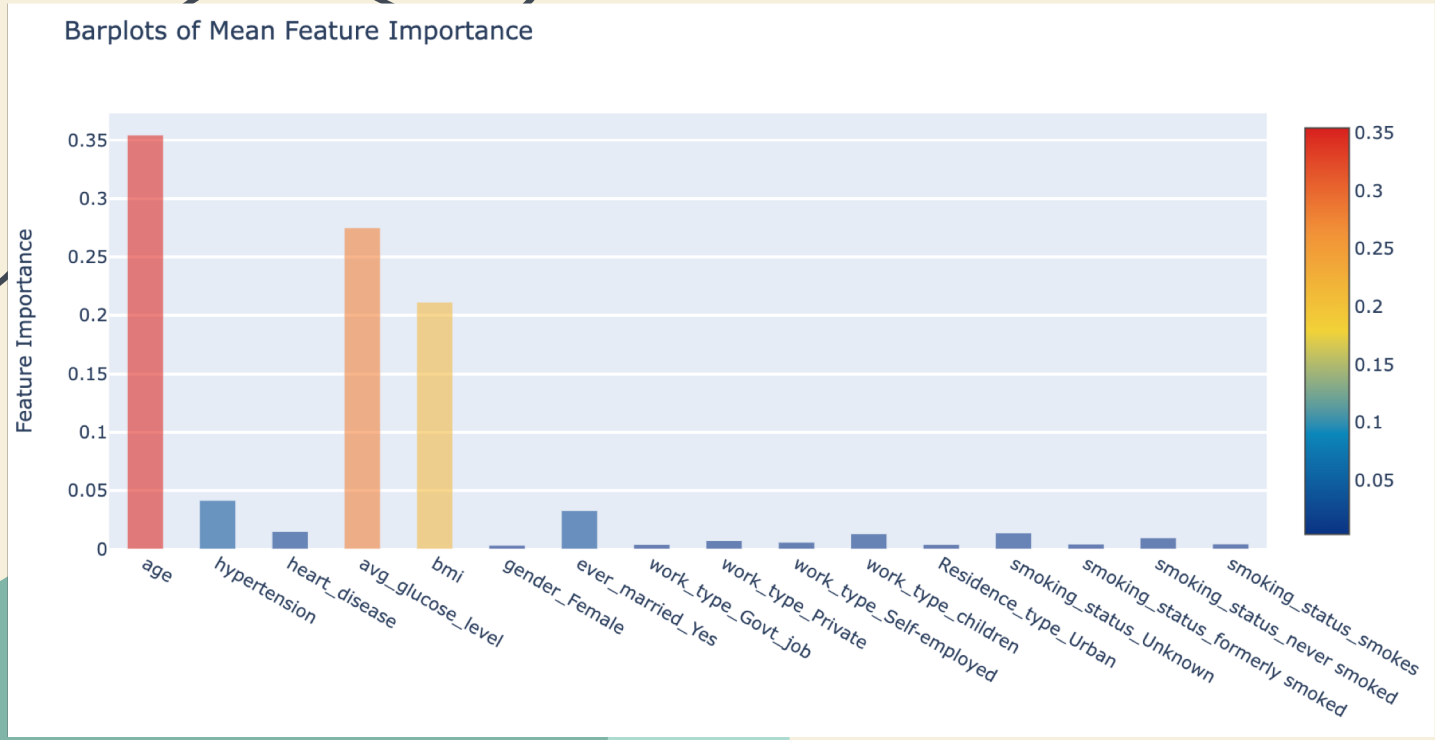
```
train, X_train, Y_train = scale_dataset(train, oversample=True)
valid, X_valid, Y_valid = scale_dataset(valid, oversample=False)
test, X_test, Y_test = scale_dataset(test, oversample=False)
```

Training the Models

- 8 models were trained
- Naive Bayes, XGBoost, Decision Tree & KNN have accuracy of more than 90%

	Algorithm	Accuracy Score
1	Naive Bayes	0.954175
7	XGBoost	0.941955
4	Decision Tree	0.922607
0	KNN	0.908350
3	SVM	0.813646
6	AdaBoost	0.801426
5	Randon Forest	0.752546
2	Logistic Regression	0.742363

Feature Importance



	age	avg_glucose_level	bmi	hypertension	ever_married_Yes	stroke
0	67.0	228.69	36.6	0	1	1
2	80.0	105.92	32.5	0	1	1
3	49.0	171.23	34.4	0	1	1
4	79.0	174.12	24.0	1	1	1
5	81.0	186.21	29.0	0	1	1
...	...	...	...	...	...	...
5104	13.0	103.08	18.6	0	0	0
5106	81.0	125.20	40.0	0	1	0
5107	35.0	82.99	30.6	0	1	0
5108	51.0	166.29	25.6	0	1	0
5109	44.0	85.28	26.2	0	1	0

4909 rows × 6 columns

Retraining

- Retraining was done using the dataset with only the top 5 features

Retraining the models

- Retraining was done using the top 5 features and top 4 models

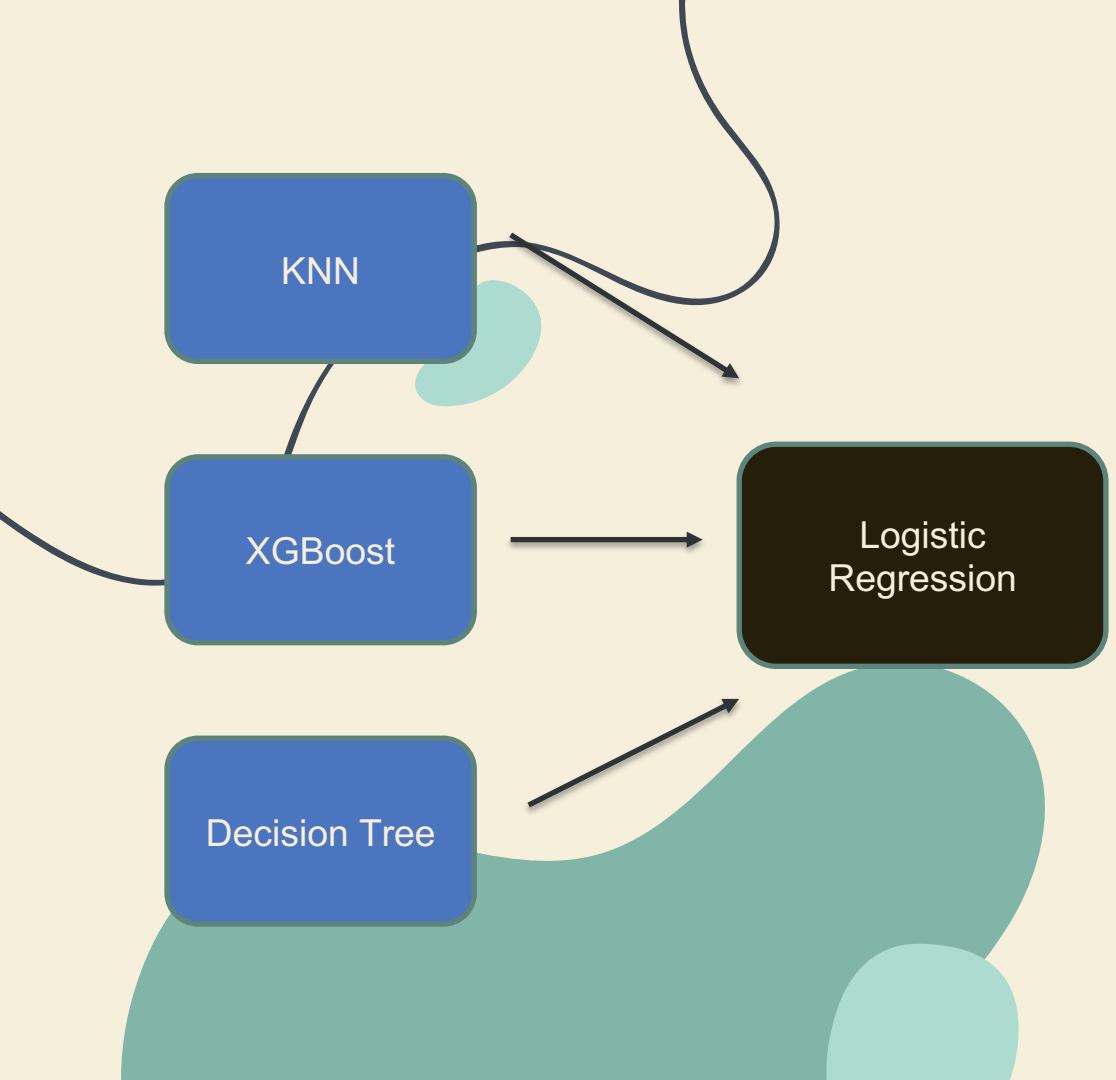
	Algorithm	Accuracy Score
1	Naive Bayes	0.954175
7	XGBoost	0.941955
4	Decision Tree	0.922607
0	KNN	0.908350
3	SVM	0.813646
6	AdaBoost	0.801426
5	Randon Forest	0.752546
2	Logistic Regression	0.742363

Before

	Algorithm	Accuracy Score
3	XGBoost	0.939919
2	Decision Tree	0.930754
0	KNN	0.906314
1	Naive Bayes	0.706721

After

Stacking



	Accuracy
knn	0.906314
xgb	0.939919
dt	0.930754
stack	1.000000

The ensemble stacking model achieves the highest accuracy than any other model taken alone.

Hyperparameter tuning - GridSearchCV

```
GridSearchCV(cv=3,
             estimator=Pipeline(steps=[('classifier', KNeighborsClassifier())]),
             n_jobs=-1,
             param_grid=[{'classifier': [KNeighborsClassifier()],
                          'classifier__n_neighbors': [1, 3, 5]},
                         {'classifier': [GradientBoostingClassifier()],
                          'classifier__max_depth': [5, 10, 20],
                          'classifier__n_estimators': [10, 50, 100, 250]},
                         {'classifier': [DecisionTreeClassifier(max_depth=5,
                                                                min_samples_split=10)],
                          'classifier__class_weight': [None, {0: 1, 1: 5},
                                                       {0: 1, 1: 10},
                                                       {0: 1, 1: 25}],
                          'classifier__max_depth': [5, 10, 25, None],
                          'classifier__min_samples_split': [2, 5, 10]}],
             scoring='accuracy')
```

```
{'classifier': DecisionTreeClassifier(max_depth=5, min_samples_split=10),
 'classifier__class_weight': None,
 'classifier__max_depth': 5,
 'classifier__min_samples_split': 10}
```

Best score = 0.9419719120317923

Hyperparameter tuning - RandomSearchCV

```
RandomizedSearchCV(cv=3,
                   estimator=Pipeline(steps=[('classifier',
                                              KNeighborsClassifier())]),
                   n_jobs=-1,
                   param_distributions=[{'classifier': [KNeighborsClassifier()],
                                         'classifier__n_neighbors': [1, 3, 5]},
                                        {'classifier': [GradientBoostingClassifier(max_depth=5,
                                                                                   n_estimators=10)],
                                         'classifier__max_depth': [5, 10, 20],
                                         'classifier__n_estimators': [10, 50,
                                                                      100,
                                                                      250]},
                                        {'classifier': [DecisionTreeClassifier(max_depth=5,
                                                                               min_samples_split=10)],
                                         'classifier__class_weight': [None,
                                                                      {0: 1,
                                                                       1: 5},
                                                                      {0: 1,
                                                                       1: 10},
                                                                      {0: 1,
                                                                       1: 25}],
                                         'classifier__max_depth': [5, 10, 25,
                                                                   None],
                                         'classifier__min_samples_split': [2, 5,
                                                                           10]}],
                   scoring='accuracy')
```

```
{'classifier__n_estimators': 10,
 'classifier__max_depth': 5,
 'classifier': GradientBoostingClassifier(max_depth=5, n_estimators=10)}
```

Best score = 0.9319708930487374

Thank You

Github link