DATE : __/__/__
PAGE : _____

HOW TO DEAL WITH A ML PROJECT

① IMPORT LIBRARIES

```
import pandas as pd
 "   numpy as np

import matplotlib.pyplot as plt  } For visualisation
 "   seaborn as sns

import warning
warnings.filter warnings('ignore') } To remove warnings

from sklearn.decomposition import PCA  } To convert high dimensional
 " sklearn.manifold import TSNE          data to low
import catboost as cb                    dimensional

from sklearn.model_selection import RandomizedSearchCV
                                  ↳ For cross validation and
                                    improve accuracy

from sklearn.linear_model import LogisticRegression
 "  "  , svm import SVC
 "  "  , tree import DecisionTreeClassifier
 "  "  , ensemble import RandomForestClassifier
                        ↳ ML algo.

from sklearn.metrics import accuracy_score
                          ↳ To check accuracy
```

DATE :__/__/__
PAGE :_____

② LOAD DATA

train = pd.read_csv ('train.csv')
test = " " " ('test.csv')

→ Check data
↳ train.head()
test.head()
↳ train.info()
↳ train.shape
↳ train.describe()

③ DATA PREPROCESSING

→ Checking null values
* train.dropna()
* train.isnull().sum()
* train.drop(index, inplace = True)
OR
train.drop(col.name, axis = 0/1)
* train[col] = train[col].fillna(Tofill)
* train[col] = np.where(np.isnan, tofill, train[col])

→ Checking duplicates

* sum(train.duplicated())
* train.drop_duplicates(keep='first', inplace = True)

DATE : __/__/__
PAGE :

④ ANALYSING DATA / DATA WRANGLING

* Labelling of data

eg Sex: M | F
↓
0 | 1

use]

```
from sklearn.preprocessing import LabelEncoder
le= LabelEncoder()
train['Sex'] = le.fit_transform(train['Sex'])
```

* Use graphs for analysing

→ Check distribution (should be normal)

```
sns.displot(x=col, data=train, kde=True)
```

* To plot multiple graphs in a single

```
features = ['Col1', 'Col2', 'Col3' ----]
for i in features
    sns.displot(x=i, data=train, kde=True)
```

OR

```
features= ['Col1', 'Col2', 'Col3' ----]
l= list(enumerate(features))   # It will give index
for i in l:                    rows, columns
    plt.subplot(3,3, i[0]+1)
    sns.scatterplot(x=i[1], y='Age', data=train)
```

Here i[0] = index
i[1] = features

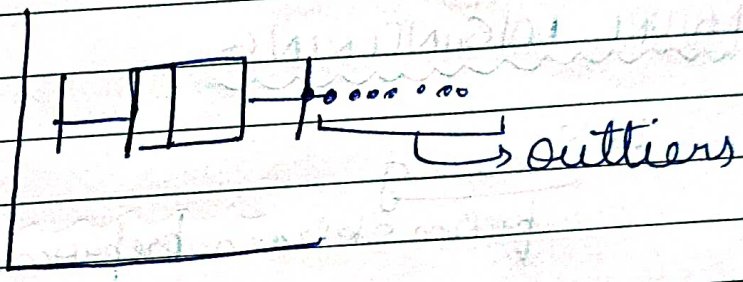
DATE : __/__/__
PAGE : _____

* To change figure size use

plt.figure (figsize = (13,2))

* Use Boxplot to check outliers

sns.boxplot (x = col, data = train)



OR

sns.boxplot (data = train)

* Use z-score or quantile (stand.dev) to remove outliers

z-score = (df.columnname - df.col.mean) / df.col.std

std → standard deviation

then remove all z-score having value lessthan -3 morethan +3

OR

```
q = train['col'].quantile(.97)   #(Change accordingly)
train = train[train['col'] < q]
```

* Check correlation of parameters

sns.heatmap(train.corr(), annot=True)

↳ Check values w.r.t. output column

*

(5) FEATURE ENGINEERING

↳ Scaling

```
from sklearn.preprocessing import
        StandardScaler
        OR
        MinMaxScaler
x = pd.dataframe(MinMaxScaler().fit_tr(x))
```

↳ Correlation

↳ heatmap

↳ Using PCA / TSNE

```
x = train.drop(['Outputcol'], axis=1)
pca = PCA(n_components=2, random_state=0)
    .fit_transform(x)

sns.scatterplot(x=pca[:,0], y=pca[:,1],
                hue=train['Outcome'])
```

Play with TSNE

DATE : __/__/__
PAGE : _____

⑥ ML Modelling

* Splitting of data

```
xtrain = train.drop(['Output col'], axis=1)
ytrain = train['Output col']

xtest = test.drop(['Output (if any)'], axis=1)
```

OR

remove any unwanted columns
↳ check by correlations

* Apply models using hyperparameters & cross validation

* For logistic regression

```
parameters = {'max_iter': [100,200,300]}
lr = LogisticRegression()
lr_rs = RandomizedSearchCV(lr, param_dist
                           = parameters,
                           cv=5, randomstate
                           =42)
lr_rs.fit(xtrain, ytrain)

y_pred = lr_rs.predict(x_test)
```

DATE : __/__/__
PAGE : ____

* For Kernel SVM

```
parameters = {
        'kernel': ['linear', 'rbf', 'poly', 'sigmoid'],
        'C': [100,5]
    }
#same
```

* For Deasion tree

```
parameters = {
        'max_depth' = np.arange(2,10,2)
    }
```

* For Randomforest

```
parameiss = {
        'n_estimators': np.arange(20,100,10),
        'max_depth': np.arange(2,16,2)
    }
```

OR To increse accuracy use Catboost model

```
train_data = cb.pool(x_train, ytrain)
model = cb.CatBoostRegressor(loss function='RMSE')

#grid search
grid = {'iteration': [100,500],
        'learning_rate': [0.03, 0.1],
        'depth': [2,4,6,8]}
model.grid-Search(grid, train-data)
pred = model.predict(x_test)
```

DATE : __/__/__
PAGE : ____

~~print (RMSE)~~ (≈0) ✓ good model

Extra Info

* Date time application

eg Date → 9/26/2009

↓ Splitting

↓

```
df['Year'] = df['Date'].apply(lamda
                x: x.split('/')[2])
```

Ily for month, day

* also you can use

```
df['Year'] = pd.to_datetime(df['Date'])
                .dt.year
```

↓

use for time, hour, min, sec, day, month etc