Classification Assignment-

1. Usually a proper way to approach a dataset is by identifying the three stages of AI Prediction

1.Identify the Problem statement

There are three stages of AI Prediction

Stage 1:

Domain Selection: Machine Learning

Stage 2:

Learning: Supervised

Stage 3:

Classification: Classification

2.CKD.csv dataset contains 399 rows × 25 columns

```
In [6]: dataset
Out[6]:
```

k to scroll output; double click to hide

	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	pcv	wc	rc	htn	dm	cad	appet	pe
0	2.000000	76.459948	c	3.0	0.0	normal	abnormal	notpresent	notpresent	148.112676	...	38.868902	8408.191126	4.705597	no	no	no	yes	yes
1	3.000000	76.459948	c	2.0	0.0	normal	normal	notpresent	notpresent	148.112676	...	34.000000	12300.000000	4.705597	no	no	no	yes	poor
2	4.000000	76.459948	a	1.0	0.0	normal	normal	notpresent	notpresent	99.000000	...	34.000000	8408.191126	4.705597	no	no	no	yes	poor
3	5.000000	76.459948	d	1.0	0.0	normal	normal	notpresent	notpresent	148.112676	...	38.868902	8408.191126	4.705597	no	no	no	yes	poor
4	5.000000	50.000000	c	0.0	0.0	normal	normal	notpresent	notpresent	148.112676	...	36.000000	12400.000000	4.705597	no	no	no	yes	poor
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
394	51.492308	70.000000	a	0.0	0.0	normal	normal	notpresent	notpresent	219.000000	...	37.000000	9800.000000	4.400000	no	no	no	yes	poor
395	51.492308	70.000000	c	0.0	2.0	normal	normal	notpresent	notpresent	220.000000	...	27.000000	8408.191126	4.705597	yes	yes	no	yes	poor
396	51.492308	70.000000	c	3.0	0.0	normal	normal	notpresent	notpresent	110.000000	...	26.000000	9200.000000	3.400000	yes	yes	no	poor	poor
397	51.492308	90.000000	a	0.0	0.0	normal	normal	notpresent	notpresent	207.000000	...	38.868902	8408.191126	4.705597	yes	yes	no	yes	poor
398	51.492308	80.000000	a	0.0	0.0	normal	normal	notpresent	notpresent	100.000000	...	53.000000	8500.000000	4.900000	no	no	no	yes	poor

399 rows × 25 columns

3. Prepocessing Method

CKD.csv has categorial column which is other wise called as (Label Encoding) data thus we are using get_ dummies function

4. I have implemented each algorithm separately in order to find the evaluation metric using Confusion Matrix for Classification.

1. RF _ Grid _ using _Classification:

[13]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier

param_grid={'criterion':['gini', 'entropy', 'log_loss'],
            'max_features':['auto','sqrt','log2'],'n_estimators':[10,100]}


grid = GridSearchCV(RandomForestClassifier(), param_grid, refit = True, verbose = 3,n_jobs=-1,scoring='f1_weighted')



# fitting the model for grid search
grid.fit(x_train, y_train)
```

```
Fitting 5 folds for each of 18 candidates, totalling 90 fits
```

```
C:\Users\sindhiya maria\anaconda3\lib\site-packages\sklearn\model_selection\_search.py:922: UserWarning: One or more of the tes
t scores are non-finite: [0.97749454 0.98480061 0.958686   0.97341245 0.95146756 0.98095962
 0.98481138 0.98111443 0.96229814 0.98104401 0.95863271 0.97725344
        nan        nan        nan        nan        nan        nan]
  warnings.warn(
C:\Users\sindhiya maria\anaconda3\lib\site-packages\sklearn\model_selection\_search.py:880: DataConversionWarning: A column-vec
tor y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  self.best_estimator_.fit(X, y, **fit_params)
```

t[13]:
```
GridSearchCV(estimator=RandomForestClassifier(), n_jobs=-1,
```

```
grid = GridSearchCV(RandomForestClassifier(), param_grid, refit = True, verbose = 3,n_jobs=-1,scoring='f1_weighted')



# fitting the model for grid search
grid.fit(x_train, y_train)
```

```
Fitting 5 folds for each of 18 candidates, totalling 90 fits
```

```
C:\Users\sindhiya maria\anaconda3\lib\site-packages\sklearn\model_selection\_search.py:922: UserWarning: One or more of the tes
t scores are non-finite: [0.97749454 0.98480061 0.958686   0.97341245 0.95146756 0.98095962
 0.98481138 0.98111443 0.96229814 0.98104401 0.95863271 0.97725344
        nan        nan        nan        nan        nan        nan]
  warnings.warn(
C:\Users\sindhiya maria\anaconda3\lib\site-packages\sklearn\model_selection\_search.py:880: DataConversionWarning: A column-vec
tor y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  self.best_estimator_.fit(X, y, **fit_params)
```

Out[13]:
```
GridSearchCV(estimator=RandomForestClassifier(), n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy', 'log_loss'],
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'n_estimators': [10, 100]},
             scoring='f1_weighted', verbose=3)
```

```
# print classification report
from sklearn.metrics  import classification_report
clf_report = classification_report(y_test, grid_predictions)
```

```
from sklearn.metrics import f1_score
f1_macro=f1_score(y_test,grid_predictions,average='weighted')
print("The f1_macro value for best parameter {}:".format(grid.best_params_),f1_macro)
```

```
The f1_macro value for best parameter {'criterion': 'entropy', 'max_features': 'auto', 'n_estimators': 10}: 0.9850141736106648
```

```
print("The confusion Matrix:\n",cm)
```

```
The confusion Matrix:
 [[51  0]
 [ 2 80]]
```

```
print("The Report:\n",clf_report)
```

```
The Report:
              precision    recall  f1-score   support

           0       0.96      1.00      0.98        51
           1       1.00      0.98      0.99        82

    accuracy                           0.98       133
   macro avg       0.98      0.99      0.98       133
```

```
In [18]: from sklearn.metrics import roc_auc_score
         roc_auc_score(y_test,grid.predict_proba(x_test)[:,1])
Out[18]: 0.9996413199426111

In [19]: table =pd.DataFrame.from_dict(re)

In [20]: table
```

Out[20]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_criterion	param_max_features	param_n_estimators	params	split0_test_score	s
0	0.022704	0.000869	0.003087	1.730212e-04	gini	auto	10	{'criterion': 'gini', 'max_features': 'auto', ...	0.963284	
1	0.177766	0.010295	0.012903	1.017577e-03	gini	auto	100	{'criterion': 'gini', 'max_features': 'auto', ...	1.000000	
2	0.021000	0.001788	0.002800	1.165321e-03	gini	sqrt	10	{'criterion': 'gini', 'max_features': 'sqrt', ...	0.981569	
3	0.171372	0.003480	0.012599	4.920636e-04	gini	sqrt	100	{'criterion': 'gini', 'max_features': 'sqrt', ...	1.000000	

1 . Random Forest best Confusion_ Matrix
The confusion Matrix:
[[51 0]
[2 80]]

The f1_macro value for best parameter {'criterion': 'entropy', 'max_features': 'auto', 'n_estimators': 10} :
0.9850141736106648
Roc_ auc _score for Random Forest (0.9996413199426111)

2. SVM _Grid _using _Classification:

```
x_test=sc.transform(x_test)

In [9]: from sklearn.svm import SVC
        # Initialize SVC with probability=True
        svc = SVC(probability=True)

        from sklearn.model_selection import GridSearchCV

        param_grid={'kernel':['rbf','poly','linear','sigmoid'],
                    'C':[10,100,1000,2000,3000],'gamma':['scale', 'auto']}


        grid = GridSearchCV(svc, param_grid, refit = True, verbose = 3,n_jobs=-1,scoring='f1_weighted')

        # fitting the model for grid search
        grid.fit(x_train, y_train)

        Fitting 5 folds for each of 40 candidates, totalling 200 fits

        C:\Users\sindhiya maria\anaconda3\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWarning: A column-vector y wa
        s passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
          return f(*args, **kwargs)

Out[9]: GridSearchCV(estimator=SVC(probability=True), n_jobs=-1,
                     param_grid={'C': [10, 100, 1000, 2000, 3000],
                                 'gamma': ['scale', 'auto'],
                                 'kernel': ['rbf', 'poly', 'linear', 'sigmoid']},
                     scoring='f1_weighted', verbose=3)
```

```
In [10]: # print best parameter after tuning
re=grid.cv_results_
grid_predictions = grid.predict(x_test)


from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,grid_predictions)

# print classification report
from sklearn.metrics  import classification_report
clf_report = classification_report(y_test, grid_predictions)

In [11]:
from sklearn.metrics import f1_score
f1_macro=f1_score(y_test,grid_predictions,average='weighted')
print("The f1_macro value for best parameter {}:".format(grid.best_params_),f1_macro)

The f1_macro value for best parameter {'C': 10, 'gamma': 'auto', 'kernel': 'poly'}: 0.955283779067923

In [12]: print("The confusion Matrix:\n",cm)

The confusion Matrix:
 [[51  0]
 [ 6 76]]

In [13]: print("The Report:\n",clf_report)

The Report:
```

```
In [13]: print("The Report:\n",clf_report)

The Report:
              precision    recall  f1-score   support

           0       0.89      1.00      0.94        51
           1       1.00      0.93      0.96        82

    accuracy                           0.95       133
   macro avg       0.95      0.96      0.95       133
weighted avg       0.96      0.95      0.96       133

In [14]: from sklearn.metrics import roc_auc_score
roc_auc_score(y_test,grid.predict_proba(x_test)[:,1])
Out[14]: 1.0

In [15]: table =pd.DataFrame.from_dict(re)

In [16]: table
Out[16]:
```

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_C	param_gamma	param_kernel	params	split0_test_score	split1_test_score	split
0	0.021083	0.008554	0.002836	4.236391e-04	10	scale	rbf	{'C': 10, 'gamma': 'scale',	0.908877	0.981014	

2. SVM_ Grid Confusion _Matrix:

The confusion Matrix:
[[51 0]
[6 76]]

The f1_macro value for best parameter {'C': 10, 'gamma': 'auto', 'kernel': 'poly'}: 0.955283779067923

roc_ auc _score for SVM (1.0)

3. DT _Grid Algorithm Classification:

```
x_test=sc.transform(x_test)
```

In [9]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV

# Define the hyperparameter grid
param_grid = {
    'criterion': ['gini', 'entropy'],
    'splitter': ['best', 'random'],
    'max_features': ['auto','sqrt','log2']}

# Create the GridSearchCV object
grid = GridSearchCV(DecisionTreeClassifier(), param_grid, refit=True, verbose=3, n_jobs=-1, scoring='f1_weighted')

# Fit the model
grid.fit(x_train, y_train)
```

```
Fitting 5 folds for each of 12 candidates, totalling 60 fits
```

Out[9]:
```
GridSearchCV(estimator=DecisionTreeClassifier(), n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'splitter': ['best', 'random']},
             scoring='f1_weighted', verbose=3)
```

In [10]:
```
# print best parameter after tuning
```

In [10]:
```
# print best parameter after tuning
re=grid.cv_results_
grid_predictions = grid.predict(x_test)


from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,grid_predictions)

# print classification report
from sklearn.metrics  import classification_report
clf_report = classification_report(y_test, grid_predictions)
```

In [11]:
```
from sklearn.metrics import f1_score
f1_macro=f1_score(y_test,grid_predictions,average='weighted')
print("The f1_macro value for best parameter {}:".format(grid.best_params_),f1_macro)
```

```
The f1_macro value for best parameter {'criterion': 'entropy', 'max_features': 'sqrt', 'splitter': 'best'}: 0.9699248120300752
```

In [12]:
```
print("The confusion Matrix:\n",cm)
```

```
The confusion Matrix:
 [[49  2]
 [ 2 80]]
```

In [13]:
```
print("The Report:\n",clf_report)
```

In [13]:
```
print("The Report:\n",clf_report)
```

```
The Report:
              precision    recall  f1-score   support

           0       0.96      0.96      0.96        51
           1       0.98      0.98      0.98        82

    accuracy                           0.97       133
   macro avg       0.97      0.97      0.97       133
weighted avg       0.97      0.97      0.97       133
```

In [14]:
```
from sklearn.metrics import roc_auc_score
roc_auc_score(y_test,grid.predict_proba(x_test)[:,1])
```

Out[14]:
```
0.9681970349115256
```

In [15]:
```
table =pd.DataFrame.from_dict(re)
```

In [16]:
```
table
```

Out[16]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_criterion	param_max_features	param_splitter	params	split0_test_score	split1_t
0	0.006400	4.408879e-03	0.002600	0.001019	gini	auto	best	{'criterion': 'gini', 'max_features': 'auto', ...	0.945100	

3.Decision_Tree_Grid Algorithm

The confusion Matrix

[[49 2]
[2 80]]

The f1_macro value for best parameter {'criterion': 'entropy', 'max_features': 'sqrt', 'splitter': 'best'}: 0.969924812 0300752

Roc _auc _score Value for Decision _Tree (0.9681970349115256)

4. Logistic _ Grid _Algorithm _Classification:

```
sc = StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.transform(x_test)
```

In [11]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV



# Define the hyperparameter grid
param_grid = {
     'solver': ['newton-cg', 'lbfgs', 'sag', 'newton-cholesky'], 'penalty': ['l2'], 'class_weight': ['balanced', 'weighted'], 'ma
    'solver': ['liblinear'], 'penalty': ['l1', 'l2'], 'class_weight': ['balanced', 'weighted'], 'max_iter': [100],
    'solver': ['saga'], 'penalty': ['l1', 'l2', 'elasticnet'], 'class_weight': ['balanced', 'weighted'], 'max_iter': [100]

     }


# Create the GridSearchCV object
grid = GridSearchCV(LogisticRegression(), param_grid, refit=True, verbose=3, n_jobs=-1, scoring='f1_weighted')

# Fit the model
grid.fit(x_train, y_train)
```

```
Fitting 5 folds for each of 6 candidates, totalling 30 fits
```

```
# Create the GridSearchCV object
grid = GridSearchCV(LogisticRegression(), param_grid, refit=True, verbose=3, n_jobs=-1, scoring='f1_weighted')

# Fit the model
grid.fit(x_train, y_train)
```

```
Fitting 5 folds for each of 6 candidates, totalling 30 fits
C:\Users\sindhiya maria\anaconda3\lib\site-packages\sklearn\model_selection\_search.py:922: UserWarning: One or more of the tes
t scores are non-finite: [0.95939983 0.9631532         nan 0.96307878 0.97060139        nan]
  warnings.warn(
C:\Users\sindhiya maria\anaconda3\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWarning: A column-vector y wa
s passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  return f(*args, **kwargs)
C:\Users\sindhiya maria\anaconda3\lib\site-packages\sklearn\linear_model\_sag.py:328: ConvergenceWarning: The max_iter was reac
hed which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
```

1]:
```
GridSearchCV(estimator=LogisticRegression(), n_jobs=-1,
             param_grid={'class_weight': ['balanced', 'weighted'],
                         'max_iter': [100],
                         'penalty': ['l1', 'l2', 'elasticnet'],
                         'solver': ['saga']},
             scoring='f1_weighted', verbose=3)
```

```
# print classification report
from sklearn.metrics  import classification_report
clf_report = classification_report(y_test, grid_predictions)
```

In [13]:
```
from sklearn.metrics import f1_score
f1_macro=f1_score(y_test,grid_predictions,average='weighted')
print("The f1_macro value for best parameter {}:".format(grid.best_params_),f1_macro)
```

```
The f1_macro value for best parameter {'class_weight': 'weighted', 'max_iter': 100, 'penalty': 'l2', 'solver': 'saga'}: 0.97011
63285572423
```

In [14]:
```
print("The confusion Matrix:\n",cm)
```

```
The confusion Matrix:
 [[51  0]
 [ 4 78]]
```

In [15]:
```
print("The Report:\n",clf_report)
```

```
The Report:
              precision    recall  f1-score   support

           0       0.93      1.00      0.96        51
           1       1.00      0.95      0.97        82

    accuracy                           0.97       133
```

```
The Report:
              precision    recall  f1-score   support

           0       0.93      1.00      0.96        51
           1       1.00      0.95      0.97        82

    accuracy                           0.97       133
   macro avg       0.96      0.98      0.97       133
weighted avg       0.97      0.97      0.97       133
```

[16]:
```
from sklearn.metrics import roc_auc_score
roc_auc_score(y_test,grid.predict_proba(x_test)[:,1])
```

t[16]:
```
0.9995217599234816
```

[17]:
```
table =pd.DataFrame.from_dict(re)
```

[18]:
```
table
```

t[18]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_class_weight	param_max_iter	param_penalty	param_solver	params	split0_test_
0	0.014603	0.001854	0.001600	0.000490	balanced	100	l1	saga	{'class_weight': 'balanced', 'max_iter': 100, ...	0.8
									{'class_weight': 'balanced'	

4. Logistic _Grid _Algorithm:

The confusion Matrix:
[[51 0]
[4 78]]

The f1_macro value for best parameter {'class _weight': 'weighted', 'max _iter': 100, 'penalty': 'l2', 'solver': ' saga'}: 0.9701163285572423

Roc _auc_ score for Logistic _Grid 0.9995217599234816

Results:

The final Machine Learning Best Model of Classification for CKD.csv dataset

By implementing algorithm in order to find the best model. I found that both Random Forest and Logistic_Algorithm has best Confusion Matrix and Roc _Auc_ Score

- 1 . Random Forest best Confusion _Matrix
  The confusion Matrix:
  [[51 0]
  [2 80]]

The f1_macro value for best parameter {'criterion': 'entropy', 'max _features': 'auto', 'n_ estimators': 10} : 0.9850141736106648

Roc _Auc _Score for Random Forest is (0.9996413199426111)

(or)

- 2. Logistic _Algorithm _Grid

  The confusion Matrix:
  [[51 0]
  [4 78]]

  The f1_macro value for best parameter {'class_ weight': 'weighted', 'max_ iter ': 100, 'penalty': 'l2', 'solver': 'saga'}: 0.9701163285572423

  Roc_ Auc _Score for Logistic _ Algorithm _Grid is 0.9995217599234816