Final Val Method 23/03/2024

GRID SEARCH CV

1. The Client Requirement Input is provides the Data Set of Insurance and Output is he wants to predict the Insurance charges based on the several parameters.

2. The Basic Information is,

Input →Dataset, Output →Insurance charges

Total No of Rows= 1338

Total No of Columns= 6

3. Here the Preprocessing Method is, to handle Categorical column using, Converting String to Number (Nominal Data→ One Hot Encoder)

1. Support Vector MachineGRIDCV

```
from sklearn.model_selection import GridSearchCV

from sklearn.svm import SVR

param_grid = {'kernel':['rbf','poly','sigmoid','linear'],

            'C':[10,100,1000,2000,3000],'gamma':['auto','scale']}

grid = GridSearchCV(SVR(), param_grid, refit = True, verbose = 3,n_jobs=-1)

# fitting the model for grid search

grid.fit(X_train, y_train)
```

```
The R_score value for best parameter {'C': 3000, 'gamma': 'scale', 'kernel':
'poly'}: 0.8577898390283539
```

3. Decision Tree

```
from sklearn.tree import DecisionTreeRegressor

from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeRegressor

param_grid = {'criterion':['mse','mae','friedman_mse','poisson'],

            'max_features': ['auto','sqrt','log2'],

            'splitter':['best','random']}

grid = GridSearchCV(DecisionTreeRegressor(), param_grid, refit = True, verbose = 3,n_jobs=-
1)

  # fitting the model for grid search

grid.fit(X_train,y_train)
```

```
The R_score value for best parameter {'criterion': 'friedman_mse', 'max_featu
res': 'sqrt', 'splitter': 'best'}: 0.7310420392095164
```

4. Random Forest

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor
param_grid = {'random_state': [0],
'n_estimators':[100],
            'criterion':['mse','mae','friedman_mse','poison']}
grid = GridSearchCV(RandomForestRegressor(), param_grid, refit = True, verbose = 3,n_jobs=-
1)

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

```
The R_score value for best parameter {'criterion': 'friedman_mse', 'n_estimat
ors': 100, 'random_state': 0}: 0.8595964959543749
```

5.Ada Boost Algorithm

```
from sklearn.model_selection import GridSearchCV

from sklearn.ensemble import AdaBoostRegressor

param_grid = {'random_state': [0],

            'n_estimators':[100], 'loss':['linear'],'learning_rate':[1.0]}

grid=GridSearchCV(AdaBoostRegressor(),param_grid,refit=True,verbose = 3,n_jobs=-1)
```

```
grid.fit(X_train,y_train)
```

```
The R_score value for best parameter {'learning_rate': 1.0, 'loss': 'linear',
 'n_estimators': 100, 'random_state': 0}: 0.8618083410771146
```

6.XGBoost Algorithm

```
from sklearn.model_selection import GridSearchCV
from xgboost import XGBRegressor
param_grid = {'n_estimators':[1000], 'max_depth':[7], 'eta':[0.1], 'subsample':[0.7], 'colsample_b
ytree':[0.8],'use_rmm':["true"],
                'booster':["gbtree"],'device':["cpu"],
                'verbosity':[1],'validate_parameters':["false"],'disable_default_eval_metric':["false"],
                'gamma':[0],'min_child_weight':[1],'max_delta_step':[0],'sampling_method':["unifor
m"],
                'colsample_bylevel':[1],'colsample_bynode':[1],'Lambda':[1],'alpha':[1],'tree_method
':["auto"],
                'scale_pos_weight':[1],'refresh_leaf':[1],'process_type':["default"],'grow_policy':["de
pthwise"],'max_leaves':[0],'max_bin':[256],
                'num_parallel_tree':[1],'save_period':[0],'task':["train"],'model_in':["NULL"],'model
_out':["NULL"],'model_dir':["models/"],'dump_format':["text"],
               'name_dump':["dump.txt"],'name_pred':["pred.txt"],'pred_margin':[0]}
grid = GridSearchCV(XGBRegressor(), param_grid, refit = True, verbose = 3,n_jobs=-1)

grid.fit(X_train,y_train)
```

```
The R_score value for best parameter {'Lambda': 1, 'alpha': 1, 'booster': 'gb
tree', 'colsample_bylevel': 1, 'colsample_bynode': 1, 'colsample_bytree': 0.8
, 'device': 'cpu', 'disable_default_eval_metric': 'false', 'dump_format': 'te
xt', 'eta': 0.1, 'gamma': 0, 'grow_policy': 'depthwise', 'max_bin': 256, 'max
_delta_step': 0, 'max_depth': 7, 'max_leaves': 0, 'min_child_weight': 1, 'mod
el_dir': 'models/', 'model_in': 'NULL', 'model_out': 'NULL', 'n_estimators':
1000, 'name_dump': 'dump.txt', 'name_pred': 'pred.txt', 'num_parallel_tree':
1, 'pred_margin': 0, 'process_type': 'default', 'refresh_leaf': 1, 'sampling_
method': 'uniform', 'save_period': 0, 'scale_pos_weight': 1, 'subsample': 0.7
, 'task': 'train', 'tree_method': 'auto', 'use_rmm': 'true', 'validate_parame
ters': 'false', 'verbosity': 1}: 0.832608353523225
```

7.LG Boost Algorithm

```
param_grid = {
    'boosting_type': ['gbdt'],
    'num_leaves': [31],
    'max_depth': [-1],
    'learning_rate': [0.1],
    'n_estimators': [100],
    'subsample_for_bin': [200000],
    'objective': [None],  # Assuming you meant to pass None, not 'None'
    'class_weight': [None],
    'min_split_gain': [0.0],
    'min_child_weight': [0.001],
    'min_child_samples': [20],
    'subsample': [1.0],
    'subsample_freq': [0],
    'force_row_wise': [True],  # Use Python boolean True instead of "true"
    'colsample_bytree': [1.0],
    'reg_alpha': [0.0],
    'reg_lambda': [0.0],
    'random_state': [None],
    'n_jobs': [None],  # Use None without quotes
    'importance_type': ['split']
}

grid = GridSearchCV(LGBMRegressor(), param_grid, refit = True, verbose = 3,n_jobs=-1)
grid.fit(X_train, y_train)
```

```
The R_score value for best parameter {'boosting_type': 'gbdt', 'class_weight'
: None, 'colsample_bytree': 1.0, 'force_row_wise': True, 'importance_type': '
split', 'learning_rate': 0.1, 'max_depth': -1, 'min_child_samples': 20, 'min_
child_weight': 0.001, 'min_split_gain': 0.0, 'n_estimators': 100, 'n_jobs': N
one, 'num_leaves': 31, 'objective': None, 'random_state': None, 'reg_alpha':
0.0, 'reg_lambda': 0.0, 'subsample': 1.0, 'subsample_for_bin': 200000, 'subsa
mple_freq': 0}: 0.8699321391117371
```

The Result of LG Boost Algorithm is better accuracy value 0.869 compared to all Algorithm.