Pivot table

Sns.barplot

**Cross\_val\_score**

* By default, the score computed at each CV iteration is the score method of the estimator. It is possible to change this by using the scoring parameter.

**n\_jobs**:  is used to specify how many concurrent processes or threads should be used.  If 1 is given, no joblib parallelism is used at all, which is useful for debugging. If set to -1, all CPUs are used. For n\_jobs below -1, **(n\_cpus + 1 + n\_jobs)** are used.

**verbose**: verbose = 0: showing nothing about the progress (silent), verbose > 1: more details being printed (e.g. “Fitting 3 folds for each of 8 candidates, totalling 24 fits“)

**scoring**: controls what metric GridSearchCV or cross\_val\_score apply to the estimators evaluated. A list of common cases is listed here: [3.3. Metrics and scoring: quantifying the quality of predictions — scikit-learn 1.3.2 documentation](https://scikit-learn.org/stable/modules/model_evaluation.html#scoring-parameter)

VotingClassifier

* StandardScaler
* stats.norm.fit / mu, sigma / stats.norm.pdf
* Q-Q plot (stats.probplot) / qq\_data[0][0] and qq\_data[0][1]
* Stats.linregress
* Ax.bar\_label -> label\_type
* SimpleImputer -> strategy, fill\_value
* OneHotEncoder -> handle\_unknown, sparse
* Param\_grids: A screen shot of a computer

  Description automatically generated
* KFold

**GridSearchCV**: is the process of performing hyperparameter tuning in order to determine the optimal values for a given model. There is no way to know in advance the best values for hyperparameters so ideally, we need to try all possible values to know the optimal values.

* **How does GridSearchCV calculate its best\_score\_?**

“Mean cross-validated score of the best\_estimator”

(e.g. If cv=5, the data will be split into train and test folds 5 times. The model will be fitted on *training sets* and scored on the *testing sets*. These 5 test scores are averaged to get the score. The highest score will be the *best\_score\_)*

* FunctionTransformer
* ColumnTransfromer -> remainder=’passthrough’
* Why we need np.exp() when predicting y\_pred?

A screenshot of a computer

Description automatically generated