

# Sample\_LAB2TPOT\_prompt

September 29, 2024

```
[ ]: # !pip install tpot pandas -q
```

## 0.0.1 PROMPT:

**Context:** I want you to act as an automatic machine learning (AutoML) bot using TPOT for me. I am working on a model that predicts sales. The code will be executed in Colab, so we must take care of the compatible versions between TPOT and skit-learn.

**Input:** Use the Iris dataset available in sklearn

**Output:** Please write python code to find the best classification model with the highest AUC score on the test set.

## 0.0.2 ChatGPT solution, fixed by Gemini

```
[1]: import sklearn
import tpot
```

```
[3]: # chatGPT Answer

# Import necessary libraries
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score, roc_curve
from tpot import TPOTClassifier
import numpy as np
#from sklearn.preprocessing import LabelBinarizer # This is not needed

# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# No need to binarize the output, TPOTClassifier can handle multiclass
↪classification
#lb = LabelBinarizer()
#y_binarized = lb.fit_transform(y)

# Split the dataset into training and testing sets
```

```

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
↳random_state=42)

# Initialize TPOTClassifier
tpot = TPOTClassifier(verbosity=2, scoring='roc_auc', generations=5,
↳population_size=20, random_state=42)

# Fit the TPOTClassifier
tpot.fit(X_train, y_train)

# Evaluate the model on the test set
y_test_pred = tpot.predict_proba(X_test)

# Calculate AUC score for each class
auc_scores = [roc_auc_score(y_test == i, y_test_pred[:, i]) for i in
↳range(y_test_pred.shape[1])]
print(f"AUC Scores for each class: {auc_scores}")

# Export the best model
tpot.export('best_pipeline.py')

```

Optimization Progress: 0% | 0/120 [00:00<?, ?pipeline/s]

Generation 1 - Current best internal CV score: -inf

```

-----
ValueError                                Traceback (most recent call last)
File c:\_git\github_repo\master_SistemasInteligentes\
↳venv\Lib\site-packages\tpot\base.py:817, in TPOTBase.fit(self, features,
↳target, sample_weight, groups)
    816         warnings.simplefilter("ignore")
--> 817         self._pop, _ = eaMuPlusLambda(
    818             population=self._pop,
    819             toolbox=self._toolbox,
    820             mu=self.population_size,
    821             lambda_=self._lambda,
    822             cxpb=self.crossover_rate,
    823             mutpb=self.mutation_rate,
    824             ngen=self.generations,
    825             pbar=self._pbar,
    826             halloffame=self._pareto_front,
    827             verbose=self.verbosity,
    828             per_generation_function=self._check_periodic_pipeline,
    829             log_file=self.log_file_,
    830         )
    832 # Allow for certain exceptions to signal a premature fit() cancellation

```

```

File c:\_git\github_repo\master_SistemasInteligentes\.
↳venv\Lib\site-packages\tpot\gp_deap.py:285, in eaMuPlusLambda(population,
↳toolbox, mu, lambda_, cxpb, mutpb, ngen, pbar, stats, halloffame, verbose,
↳per_generation_function, log_file)
    284 if per_generation_function is not None:
--> 285     per_generation_function(gen)
    287 # Update the statistics with the new population

File c:\_git\github_repo\master_SistemasInteligentes\.
↳venv\Lib\site-packages\tpot\base.py:1184, in TPOTBase.
↳_check_periodic_pipeline(self, gen)
    1174 """If enough time has passed, save a new optimized pipeline. Currently,
↳used in the per generation hook in the optimization loop.
    1175 Parameters
    1176 -----
    1177 (...)
    1182 None
    1183 """
-> 1184 self._update_top_pipeline()
    1185 if self.periodic_checkpoint_folder is not None:

File c:\_git\github_repo\master_SistemasInteligentes\.
↳venv\Lib\site-packages\tpot\base.py:925, in TPOTBase._update_top_pipeline(self)
    923 from sklearn.model_selection import cross_val_score
--> 925 cv_scores = cross_val_score(
    926     sklearn_pipeline,
    927     self.pretest_X,
    928     self.pretest_y,
    929     cv=self.cv,
    930     scoring=self.scoring_function,
    931     verbose=0,
    932     error_score="raise",
    933 )
    934 break

File c:\_git\github_repo\master_SistemasInteligentes\.
↳venv\Lib\site-packages\sklearn\utils\_param_validation.py:213, in
↳validate_params.<locals>.decorator.<locals>.wrapper(*args, **kwargs)
    208     with config_context(
    209         skip_parameter_validation=(
    210             prefer_skip_nested_validation or global_skip_validation
    211         )
    212     ):
--> 213         return func(*args, **kwargs)
    214 except InvalidParameterError as e:
    215     # When the function is just a wrapper around an estimator, we allow
    216     # the function to delegate validation to the estimator, but we
↳replace
    217     # the name of the estimator by the name of the function in the error

```

```

218     # message to avoid confusion.

File c:\_git\github_repo\master_SistemasInteligentes\.
↳venv\Lib\site-packages\sklearn\model_selection\_validation.py:712, in
↳cross_val_score(estimator, X, y, groups, scoring, cv, n_jobs, verbose,
↳fit_params, params, pre_dispatch, error_score)
    710 scorer = check_scoring(estimator, scoring=scoring)
--> 712 cv_results = cross_validate(
    713     estimator=estimator,
    714     X=X,
    715     y=y,
    716     groups=groups,
    717     scoring={"score": scorer},
    718     cv=cv,
    719     n_jobs=n_jobs,
    720     verbose=verbose,
    721     fit_params=fit_params,
    722     params=params,
    723     pre_dispatch=pre_dispatch,
    724     error_score=error_score,
    725 )
    726 return cv_results["test_score"]

File c:\_git\github_repo\master_SistemasInteligentes\.
↳venv\Lib\site-packages\sklearn\utils\_param_validation.py:213, in
↳validate_params.<locals>.decorator.<locals>.wrapper(*args, **kwargs)
    208     with config_context(
    209         skip_parameter_validation=(
    210             prefer_skip_nested_validation or global_skip_validation
    211         )
    212     ):
--> 213     return func(*args, **kwargs)
    214 except InvalidParameterError as e:
    215     # When the function is just a wrapper around an estimator, we allow
    216     # the function to delegate validation to the estimator, but we
↳replace
    217     # the name of the estimator by the name of the function in the error
    218     # message to avoid confusion.

File c:\_git\github_repo\master_SistemasInteligentes\.
↳venv\Lib\site-packages\sklearn\model_selection\_validation.py:423, in
↳cross_validate(estimator, X, y, groups, scoring, cv, n_jobs, verbose,
↳fit_params, params, pre_dispatch, return_train_score, return_estimator,
↳return_indices, error_score)
    422 parallel = Parallel(n_jobs=n_jobs, verbose=verbose,
↳pre_dispatch=pre_dispatch)
--> 423 results = parallel(
    424     delayed(_fit_and_score)(
    425         clone(estimator),

```

```

426     X,
427     y,
428     scorer=scorers,
429     train=train,
430     test=test,
431     verbose=verbose,
432     parameters=None,
433     fit_params=routed_params.estimator.fit,
434     score_params=routed_params.scorer.score,
435     return_train_score=return_train_score,
436     return_times=True,
437     return_estimator=return_estimator,
438     error_score=error_score,
439 )
440     for train, test in indices
441 )
443 _warn_or_raise_about_fit_failures(results, error_score)

```

```

File c:\_git\github_repo\master_SistemasInteligentes\.
↳venv\Lib\site-packages\sklearn\utils\parallel.py:74, in Parallel.
↳__call__(self, iterable)
    70 iterable_with_config = (
    71     (_with_config(delayed_func, config), args, kwargs)
    72     for delayed_func, args, kwargs in iterable
    73 )
---> 74 return super().__call__(iterable_with_config)

```

```

File c:\_git\github_repo\master_SistemasInteligentes\.
↳venv\Lib\site-packages\joblib\parallel.py:1918, in Parallel.__call__(self,
↳iterable)
    1917     next(output)
-> 1918     return output if self.return_generator else list(output)
    1920 # Let's create an ID that uniquely identifies the current call. If the
    1921 # call is interrupted early and that the same instance is immediately
    1922 # re-used, this id will be used to prevent workers that were
    1923 # concurrently finalizing a task from the previous call to run the
    1924 # callback.

```

```

File c:\_git\github_repo\master_SistemasInteligentes\.
↳venv\Lib\site-packages\joblib\parallel.py:1847, in Parallel.
↳get_sequential_output(self, iterable)
    1846 self.n_dispatched_tasks += 1
-> 1847 res = func(*args, **kwargs)
    1848 self.n_completed_tasks += 1

```

```

File c:\_git\github_repo\master_SistemasInteligentes\.
↳venv\Lib\site-packages\sklearn\utils\parallel.py:136, in _FuncWrapper.
↳__call__(self, *args, **kwargs)
    135 with config_context(**config):

```

```
--> 136     return self.function(*args, **kwargs)
```

File c:\\_git\github\_repo\master\_SistemasInteligentes\.

```
↳venv\Lib\site-packages\sklearn\model_selection\_validation.py:910, in
↳fit_and_score(estimator, X, y, scorer, train, test, verbose, parameters,
↳fit_params, score_params, return_train_score, return_parameters,
↳return_n_test_samples, return_times, return_estimator, split_progress,
↳candidate_progress, error_score)
    909 fit_time = time.time() - start_time
--> 910 test_scores = _score(
    911     estimator, X_test, y_test, scorer, score_params_test, error_score
    912 )
    913 score_time = time.time() - start_time - fit_time
```

File c:\\_git\github\_repo\master\_SistemasInteligentes\.

```
↳venv\Lib\site-packages\sklearn\model_selection\_validation.py:971, in
↳_score(estimator, X_test, y_test, scorer, score_params, error_score)
    970     else:
--> 971         scores = scorer(estimator, X_test, y_test, **score_params)
    972 except Exception:
```

File c:\\_git\github\_repo\master\_SistemasInteligentes\.

```
↳venv\Lib\site-packages\sklearn\metrics\_scorer.py:147, in _MultimetricScorer.
↳__call__(self, estimator, *args, **kwargs)
    146 if self._raise_exc:
--> 147     raise e
    148 else:
```

File c:\\_git\github\_repo\master\_SistemasInteligentes\.

```
↳venv\Lib\site-packages\sklearn\metrics\_scorer.py:139, in _MultimetricScorer.
↳__call__(self, estimator, *args, **kwargs)
    138 if isinstance(scorer, _BaseScorer):
--> 139     score = scorer._score(
    140         cached_call, estimator, *args, **routed_params.get(name).score
    141     )
    142 else:
```

File c:\\_git\github\_repo\master\_SistemasInteligentes\.

```
↳venv\Lib\site-packages\sklearn\metrics\_scorer.py:376, in _Scorer._score(self
↳method_caller, estimator, X, y_true, **kwargs)
    375 scoring_kwargs = {**self._kwargs, **kwargs}
--> 376 return self._sign * self._score_func(y_true, y_pred, **scoring_kwargs)
```

File c:\\_git\github\_repo\master\_SistemasInteligentes\.

```
↳venv\Lib\site-packages\sklearn\utils\_param_validation.py:213, in
↳validate_params.<locals>.decorator.<locals>.wrapper(*args, **kwargs)
    208     with config_context(
    209         skip_parameter_validation=(
    210             prefer_skip_nested_validation or global_skip_validation
    211         )
```

```

212     ):
--> 213         return func(*args, **kwargs)
214     except InvalidParameterError as e:
215         # When the function is just a wrapper around an estimator, we allow
216         # the function to delegate validation to the estimator, but we
↪replace
217         # the name of the estimator by the name of the function in the error
218         # message to avoid confusion.

```

File c:\\_git\github\_repo\master\_SistemasInteligentes\.

```

↪venv\Lib\site-packages\sklearn\metrics\_ranking.py:633, in
↪roc_auc_score(y_true, y_score, average, sample_weight, max_fpr, multi_class,
↪labels)
632 if multi_class == "raise":
--> 633     raise ValueError("multi_class must be in ('ovo', 'ovr')")
634 return _multiclass_roc_auc_score(
635     y_true, y_score, labels, multi_class, average, sample_weight
636 )

```

**ValueError:** multi\_class must be in ('ovo', 'ovr')

During handling of the above exception, another exception occurred:

```

ValueError                                Traceback (most recent call last)
Cell In[3], line 27
    24 tpot = TPOTClassifier(verbosity=2, scoring='roc_auc', generations=5,
↪population_size=20, random_state=42)
    26 # Fit the TPOTClassifier
--> 27 tpot.fit(X_train, y_train)
    29 # Evaluate the model on the test set
    30 y_test_pred = tpot.predict_proba(X_test)

```

File c:\\_git\github\_repo\master\_SistemasInteligentes\.

```

↪venv\Lib\site-packages\tpot\base.py:864, in TPOTBase.fit(self, features,
↪target, sample_weight, groups)
861     except (KeyboardInterrupt, SystemExit, Exception) as e:
862         # raise the exception if it's our last attempt
863         if attempt == (attempts - 1):
--> 864             raise e
865 return self

```

File c:\\_git\github\_repo\master\_SistemasInteligentes\.

```

↪venv\Lib\site-packages\tpot\base.py:855, in TPOTBase.fit(self, features,
↪target, sample_weight, groups)
852 if not isinstance(self._pbar, type(None)):
853     self._pbar.close()
--> 855 self._update_top_pipeline()
856 self._summary_of_best_pipeline(features, target)
857 # Delete the temporary cache before exiting

```

```

File c:\_git\github_repo\master_SistemasInteligentes\.
↳venv\Lib\site-packages\tpot\base.py:925, in TPOTBase._update_top_pipeline(sel
922     sklearn_pipeline = self._toolbox.compile(expr=pipeline)
923     from sklearn.model_selection import cross_val_score
--> 925     cv_scores = cross_val_score(
926         sklearn_pipeline,
927         self.pretest_X,
928         self.pretest_y,
929         cv=self.cv,
930         scoring=self.scoring_function,
931         verbose=0,
932         error_score="raise",
933     )
934     break
935 raise RuntimeError(
936     "There was an error in the TPOT optimization "
937     "process. This could be because the data was "
(...)
946     "https://epistasislab.github.io/tpot/using/"
947 )

```

```

File c:\_git\github_repo\master_SistemasInteligentes\.
↳venv\Lib\site-packages\sklearn\utils\_param_validation.py:213, in
↳validate_params.<locals>.decorator.<locals>.wrapper(*args, **kwargs)
207 try:
208     with config_context(
209         skip_parameter_validation=(
210             prefer_skip_nested_validation or global_skip_validation
211         )
212     ):
--> 213         return func(*args, **kwargs)
214 except InvalidParameterError as e:
215     # When the function is just a wrapper around an estimator, we allow
216     # the function to delegate validation to the estimator, but we
↳replace
217     # the name of the estimator by the name of the function in the error
218     # message to avoid confusion.
219     msg = re.sub(
220         r"parameter of \w+ must be",
221         f"parameter of {func.__qualname__} must be",
222         str(e),
223     )

```

```

File c:\_git\github_repo\master_SistemasInteligentes\.
↳venv\Lib\site-packages\sklearn\model_selection\_validation.py:712, in
↳cross_val_score(estimator, X, y, groups, scoring, cv, n_jobs, verbose,
↳fit_params, params, pre_dispatch, error_score)

```



```

709 # To ensure multimetric format is not supported
710 scorer = check_scoring(estimator, scoring=scoring)
--> 712 cv_results = cross_validate(
713     estimator=estimator,
714     X=X,
715     y=y,
716     groups=groups,
717     scoring={"score": scorer},
718     cv=cv,
719     n_jobs=n_jobs,
720     verbose=verbose,
721     fit_params=fit_params,
722     params=params,
723     pre_dispatch=pre_dispatch,
724     error_score=error_score,
725 )
726 return cv_results["test_score"]

```

File c:\\_git\github\_repo\master\_SistemasInteligentes\.

```

->venv\Lib\site-packages\sklearn\utils\_param_validation.py:213, in
->validate_params.<locals>.decorator.<locals>.wrapper(*args, **kwargs)
207 try:
208     with config_context(
209         skip_parameter_validation=(
210             prefer_skip_nested_validation or global_skip_validation
211         )
212     ):
--> 213     return func(*args, **kwargs)
214 except InvalidParameterError as e:
215     # When the function is just a wrapper around an estimator, we allow
216     # the function to delegate validation to the estimator, but we
->replace
217     # the name of the estimator by the name of the function in the error
218     # message to avoid confusion.
219     msg = re.sub(
220         r"parameter of \w+ must be",
221         f"parameter of {func.__qualname__} must be",
222         str(e),
223     )

```

File c:\\_git\github\_repo\master\_SistemasInteligentes\.

```

->venv\Lib\site-packages\sklearn\model_selection\_validation.py:423, in
->cross_validate(estimator, X, y, groups, scoring, cv, n_jobs, verbose,
->fit_params, params, pre_dispatch, return_train_score, return_estimator,
->return_indices, error_score)
420 # We clone the estimator to make sure that all the folds are
421 # independent, and that it is pickle-able.
422 parallel = Parallel(n_jobs=n_jobs, verbose=verbose,
->pre_dispatch=pre_dispatch)

```

```

--> 423 results = parallel(
424     delayed(_fit_and_score)(
425         clone(estimator),
426         X,
427         y,
428         scorer=scorers,
429         train=train,
430         test=test,
431         verbose=verbose,
432         parameters=None,
433         fit_params=routed_params.estimator.fit,
434         score_params=routed_params.scorer.score,
435         return_train_score=return_train_score,
436         return_times=True,
437         return_estimator=return_estimator,
438         error_score=error_score,
439     )
440     for train, test in indices
441 )
443 _warn_or_raise_about_fit_failures(results, error_score)
445 # For callable scoring, the return type is only know after calling. If
↳the
446 # return type is a dictionary, the error scores can now be inserted wit
447 # the correct key.

```

File c:\\_git\github\_repo\master\_SistemasInteligentes\.

```

↳venv\Lib\site-packages\sklearn\utils\parallel.py:74, in Parallel.
↳__call__(self, iterable)

```

```

69 config = get_config()
70 iterable_with_config = (
71     (_with_config(delayed_func, config), args, kwargs)
72     for delayed_func, args, kwargs in iterable
73 )
---> 74 return super().__call__(iterable_with_config)

```

File c:\\_git\github\_repo\master\_SistemasInteligentes\.

```

↳venv\Lib\site-packages\joblib\parallel.py:1918, in Parallel.__call__(self,
↳iterable)

```

```

1916 output = self._get_sequential_output(iterable)
1917 next(output)
-> 1918 return output if self.return_generator else list(output)
1920 # Let's create an ID that uniquely identifies the current call. If the
1921 # call is interrupted early and that the same instance is immediately
1922 # re-used, this id will be used to prevent workers that were
1923 # concurrently finalizing a task from the previous call to run the
1924 # callback.
1925 with self._lock:

```

```

File c:\_git\github_repo\master_SistemasInteligentes\.
↳venv\Lib\site-packages\joblib\parallel.py:1847, in Parallel.
↳_get_sequential_output(self, iterable)
    1845 self.n_dispatched_batches += 1
    1846 self.n_dispatched_tasks += 1
-> 1847 res = func(*args, **kwargs)
    1848 self.n_completed_tasks += 1
    1849 self.print_progress()

```

```

File c:\_git\github_repo\master_SistemasInteligentes\.
↳venv\Lib\site-packages\sklearn\utils\parallel.py:136, in _FuncWrapper.
↳__call__(self, *args, **kwargs)
    134     config = {}
    135     with config_context(**config):
--> 136     return self.function(*args, **kwargs)

```

```

File c:\_git\github_repo\master_SistemasInteligentes\.
↳venv\Lib\site-packages\sklearn\model_selection\_validation.py:910, in _
↳_fit_and_score(estimator, X, y, scorer, train, test, verbose, parameters, _
↳_fit_params, score_params, return_train_score, return_parameters, _
↳_return_n_test_samples, return_times, return_estimator, split_progress, _
↳_candidate_progress, error_score)
    907 result["fit_error"] = None
    909 fit_time = time.time() - start_time
--> 910 test_scores = _score(
    911     estimator, X_test, y_test, scorer, score_params_test, error_score
    912 )
    913 score_time = time.time() - start_time - fit_time
    914 if return_train_score:

```

```

File c:\_git\github_repo\master_SistemasInteligentes\.
↳venv\Lib\site-packages\sklearn\model_selection\_validation.py:971, in _
↳_score(estimator, X_test, y_test, scorer, score_params, error_score)
    969     scores = scorer(estimator, X_test, **score_params)
    970     else:
--> 971     scores = scorer(estimator, X_test, y_test, **score_params)
    972 except Exception:
    973     if isinstance(scorer, _MultimetricScorer):
    974         # If `_MultimetricScorer` raises exception, the `error_score`
    975         # parameter is equal to "raise".

```

```

File c:\_git\github_repo\master_SistemasInteligentes\.
↳venv\Lib\site-packages\sklearn\metrics\scorer.py:147, in _MultimetricScorer.
↳__call__(self, estimator, *args, **kwargs)
    145 except Exception as e:
    146     if self._raise_exc:
--> 147     raise e
    148     else:
    149     scores[name] = format_exc()

```

```

File c:\_git\github_repo\master_SistemasInteligentes\.
↳venv\Lib\site-packages\sklearn\metrics\_scorer.py:139, in _MultimetricScorer.
↳__call__(self, estimator, *args, **kwargs)
    137 try:
    138     if isinstance(scorer, _BaseScorer):
--> 139         score = scorer._score(
    140             estimator, *args, **kwargs
    141         )
    142     else:
    143         score = scorer(estimator, *args, **kwargs)

```

```

File c:\_git\github_repo\master_SistemasInteligentes\.
↳venv\Lib\site-packages\sklearn\metrics\_scorer.py:376, in _Scorer._score(self, estimator, X, y_true, **kwargs)
    371 y_pred = method_caller(
    372     estimator, response_method.__name__, X, pos_label=pos_label
    373 )
    375 scoring_kwargs = {**self._kwargs, **kwargs}
--> 376 return self._sign * self._score_func(y_true, y_pred, **scoring_kwargs)

```

```

File c:\_git\github_repo\master_SistemasInteligentes\.
↳venv\Lib\site-packages\sklearn\utils\_param_validation.py:213, in _validate_params
    207 try:
    208     with config_context(
    209         skip_parameter_validation=(
    210             prefer_skip_nested_validation or global_skip_validation
    211         )
    212     ):
--> 213         return func(*args, **kwargs)
    214 except InvalidParameterError as e:
    215     # When the function is just a wrapper around an estimator, we allow
    216     # the function to delegate validation to the estimator, but we
    217     # replace
    218     # the name of the estimator by the name of the function in the error
    219     # message to avoid confusion.
    220     msg = re.sub(
    221         r"parameter of \w+ must be",
    222         f"parameter of {func.__qualname__} must be",
    223         str(e),
    224     )

```

```

File c:\_git\github_repo\master_SistemasInteligentes\.
↳venv\Lib\site-packages\sklearn\metrics\_ranking.py:633, in _roc_auc_score(y_true, y_score, average, sample_weight, max_fpr, multi_class, labels)
    626     raise ValueError(
    627         "Partial AUC computation not available in "

```

```

628         "multiclass setting, 'max_fpr' must be"
629         " set to `None`, received `max_fpr={0}` "
630         "instead".format(max_fpr)
631     )
632     if multi_class == "raise":
--> 633         raise ValueError("multi_class must be in ('ovo', 'ovr')")
634     return _multiclass_roc_auc_score(
635         y_true, y_score, labels, multi_class, average, sample_weight
636     )
637 elif y_type == "binary":

```

```

ValueError: multi_class must be in ('ovo', 'ovr')

```

The previous code produces an error `ValueError: multi_class must be in ('ovo', 'ovr')`, which the Gemini assistant solves by changing the metric by `roc_auc_ovo` and adding the parameter `multiclass='ovo'` when calculating the AUC scores.

```

[ ]: # chatGPT Answer - Repaired by Gemini

# Import necessary libraries
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score, roc_curve
from tpot import TPOTClassifier
import numpy as np
#from sklearn.preprocessing import LabelBinarizer # This is not needed

# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# No need to binarize the output, TPOTClassifier can handle multiclass
↳classification
#lb = LabelBinarizer()
#y_binarized = lb.fit_transform(y)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
↳random_state=42)

# Initialize TPOTClassifier
# Added the multi_class parameter to the scoring function
tpot = TPOTClassifier(verbosity=2, scoring= 'roc_auc_ovo', generations=5,
↳population_size=20, random_state=42)

# Fit the TPOTClassifier

```

```

tpot.fit(X_train, y_train)

# Evaluate the model on the test set
y_test_pred = tpot.predict_proba(X_test)

# Calculate AUC score for each class
# Added the multi_class parameter to the scoring function
auc_scores = [roc_auc_score(y_test == i, y_test_pred[:, i], multi_class='ovo')
               ↪for i in range(y_test_pred.shape[1])]
print(f"AUC Scores for each class: {auc_scores}")

# Export the best model
tpot.export('best_pipeline.py')

```

Optimization Progress: 0% | 0/120 [00:00<?, ?pipeline/s]

Generation 1 - Current best internal CV score: 1.0

Generation 2 - Current best internal CV score: 1.0

Generation 3 - Current best internal CV score: 1.0

Generation 4 - Current best internal CV score: 1.0

Generation 5 - Current best internal CV score: 1.0

Best pipeline: MLPClassifier(input\_matrix, alpha=0.0001,  
learning\_rate\_init=0.001)

AUC Scores for each class: [1.0, 1.0, 1.0]

We can check now the file with the best model (best\_pipeline.py) and predict values for new data.

```

[ ]: #best_pipeline.py

import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier

# NOTE: Make sure that the outcome column is labeled 'target' in the data file
tpot_data = pd.read_csv('PATH/TO/DATA/FILE', sep='COLUMN_SEPARATOR', dtype=np.
    ↪float64)
features = tpot_data.drop('target', axis=1)
training_features, testing_features, training_target, testing_target = \
    train_test_split(features, tpot_data['target'], random_state=42)

# Average CV score on the training set was: 1.0

```

```

exported_pipeline = MLPClassifier(alpha=0.0001, learning_rate_init=0.001)
# Fix random state in exported estimator
if hasattr(exported_pipeline, 'random_state'):
    setattr(exported_pipeline, 'random_state', 42)

exported_pipeline.fit(training_features, training_target)
results = exported_pipeline.predict(testing_features)

```

### 0.0.3 Claude AI Solution

Perfecto a la primera!

```

[ ]: from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score
from tpot import TPOTClassifier
import numpy as np

# Load the Iris dataset
iris = load_iris()
X, y = iris.data, iris.target

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    random_state=42)

# Initialize TPOT
tpot = TPOTClassifier(
    generations=5,
    population_size=20,
    cv=5,
    random_state=42,
    verbosity=2,
    scoring='roc_auc_ovr',
    n_jobs=-1
)

# Fit TPOT
tpot.fit(X_train, y_train)

# Get the best model
best_model = tpot.fitted_pipeline_

# Make predictions on the test set
y_pred_proba = best_model.predict_proba(X_test)

# Calculate the AUC score

```

```
auc_score = roc_auc_score(y_test, y_pred_proba, multi_class='ovr')

print(f"Best pipeline: {best_model}")
print(f"AUC score on test set: {auc_score:.4f}")

# Export the best pipeline as a Python script
tpot.export('tpot_iris_pipeline.py')
```

Optimization Progress: 0%| | 0/120 [00:00<?, ?pipeline/s]

Generation 1 - Current best internal CV score: 1.0

Generation 2 - Current best internal CV score: 1.0

Generation 3 - Current best internal CV score: 1.0

Generation 4 - Current best internal CV score: 1.0

Generation 5 - Current best internal CV score: 1.0

Best pipeline: MLPClassifier(input\_matrix, alpha=0.0001,  
learning\_rate\_init=0.001)

Best pipeline: Pipeline(steps=[('mlpclassifier',  
MLPClassifier(random\_state=42))])

AUC score on test set: 1.0000