```
[UNRESOLVED_COLUMN.WITH_SUGGESTION] A column, variable, or function parameter with name `_automl_sample_weight_0000` cannot be resolved. Did you mean one of the following? [`_automl_split_col_0000`, `total_late_fees_charged`, `order_value_total`, `sunday_open_ratio`, `friday_open_ratio`]. SQLSTATE: 42703
File <command-4369244694275739>, line 8
      6   from databricks.automl.classifier import Classifier
      7   classifier = Classifier(context_type=ContextType.DATABRICKS)
----> 8   classifier.fit(**kwargs)
      9 elif problem_type == "regression":
     10   from databricks.automl.regressor import Regressor
File /databricks/.python_edge_libs/databricks/automl/classifier.py:21, in Classifier.fit(self, **kwargs)
     20 def fit(self, **kwargs):
---> 21     return internal.classifier.Classifier(context_type=self._context_type).fit(**kwargs)
File /databricks/.python_edge_libs/databricks/automl/shared/instrumentation.py:210, in instrumented(func, self, args, kwargs)
    208         failure = Failure(error="unknown")
    209     _flush_event(spark=self.spark, event=failure, run_id=run_id)
--> 210     raise e
    211 else:
    212     success = Success(
    213         experiment_id=summary.experiment.experiment_id,
    214         num_trials=len(summary.trials),
   (...)
    219         is_early_stopped=summary.is_early_stopped,
    220         output_table_present=summary.output_table_name is not None)
File /databricks/.python_edge_libs/databricks/automl/shared/instrumentation.py:203, in instrumented(func, self, args, kwargs)
    200 _flush_event(spark=self.spark, event=start, run_id=run_id)
    202 try:
--> 203     summary = func(*args, **kwargs)
    204 except BaseException as e:
    205     if isinstance(e, KeyboardInterrupt) or issubclass(type(e), AutomlError):
File /databricks/.python_edge_libs/databricks/automl/internal/supervised_learner.py:289, in SupervisedLearner.fit(self, dataset, target_col, data_dir, exclude_cols, exclude_columns, exclude_frameworks, feature_store_lookups, imputers, max_trials, timeout_minutes, time_col, pos_label, experiment, home_dir, metric, parallelism, run_id, split_col, sample_weight_col)
    287     mlflow_utils.set_experiment_state(context.experiment_id, RunState.FAILED)
    288     context.set_experiment_error("An unknown error occurred")
--> 289     raise e
    290 else:
    291     _logger.info(f"AutoML run with experiment id: {context.experiment_id} succeeded.")
File /databricks/.python_edge_libs/databricks/automl/internal/supervised_learner.py:228, in SupervisedLearner.fit(self, dataset, target_col, data_dir, exclude_cols, exclude_columns, exclude_frameworks, feature_store_lookups, imputers, max_trials, timeout_minutes, time_col, pos_label, experiment, home_dir, metric, parallelism, run_id, split_col, sample_weight_col)
    215 context.set_experiment_init(
    216     target_col=target_col,
    217     data_dir=data_dir,
   (...)
    221     evaluation_metric=metric,
    222     spark_version=self._get_cluster_info().get("runtime_version"))
    224 no_semantic_type_detection_cols = list(imputers.keys())
    226 (data_source, runtime_version, preprocess_result, data_exploration_start_time, timeout,
    227  data_exp_nb_url, pos_label, split_col, sample_weight_col,
--> 228  is_sample_weight_from_user) = self._preprocess_data_impl(
    229      dataset=dataset,
    230      target_col=target_col,
    231      context=context,
    232      data_dir=data_dir,
    233      metric=metric,
    234      timeout=timeout,
    235      time_col=time_col,
    236      split_col=split_col,
    237      alert_manager=alert_manager,
    238      run_id=run_id,
    239      no_semantic_type_detection_cols=no_semantic_type_detection_cols,
    240      exclude_cols=exclude_cols,
    241      pos_label=pos_label,
    242      feature_store_lookups=feature_store_lookups,
    243      sample_weight_col=sample_weight_col)
    245 # Time check after running data exploration notebook
    246 training_start_time, timeout = time_check(data_exploration_start_time, timeout,
    247                                           alert_manager)
File /databricks/.python_edge_libs/databricks/automl/internal/supervised_learner.py:559, in SupervisedLearner._preprocess_data_impl(self, dataset, target_col, context, data_dir, metric, timeout, time_col, split_col, alert_manager, run_id, no_semantic_type_detection_cols, exclude_cols, pos_label, feature_store_lookups, sample_weight_col)
    556     intermediate_stats.supported_cols.append(sample_weight_col)
    557     is_sample_weight_from_user = False
--> 559 dataset = self._sample_if_necessary(
    560     train_df=train_df,
    561     val_df=val_df,
    562     test_df=test_df,
    563     target_col=target_col,
    564     fraction=precise_sample_fraction,
    565     intermediate_stats=intermediate_stats,
    566     alert_manager=alert_manager,
    567     sample_weight_col=sample_weight_col,
    568     is_sample_weight_from_user=is_sample_weight_from_user)
    570 sample_fraction_to_report = size_estimator_result.sample_fraction_to_report
    571 if sample_fraction_to_report:
File /databricks/.python_edge_libs/databricks/automl/internal/supervised_learner.py:1093, in SupervisedLearner._sample_if_necessary(self, train_df, val_df, test_df, target_col, fraction, intermediate_stats, alert_manager, sample_weight_col, is_sample_weight_from_user)
   1091 # Sample val / test data
   1092 if fraction:
-> 1093     val_df = self._sample(
   1094         val_df,
   1095         fraction,
   1096         target_col,
   1097         intermediate_stats,
   1098         alert_manager,
   1099         min_rows_to_ensure=ClassificationStatsCalculator.MIN_ROWS_PER_LABEL_AFTER_SPLIT,
   1100         sample_weight_col=sample_weight_col)
   1101     test_df = self._sample(
```

```
   1102         test_df,
   1103         fraction,
   (...)
   1107         min_rows_to_ensure=ClassificationStatsCalculator.MIN_ROWS_PER_LABEL_AFTER_SPLIT,
   1108         sample_weight_col=sample_weight_col)
   1110 # Keep the schema of each split the same
File /databricks/.python_edge_libs/databricks/automl/internal/classifier.py:107, in Classifier._sample(self, dataset, fraction, target_col, dataset_stats, alert_manager, min_rows_to_ensure, sample_weight_col)
    105 # implies that sample_weight_col is passed in by user
    106 if sample_weight_col:
--> 107     dataset = self._maintain_sum_of_weights(dataset, sample_weight_col, target_col,
    108                                             fractions)
    109 return dataset
File /databricks/.python_edge_libs/databricks/automl/internal/classifier.py:301, in Classifier._maintain_sum_of_weights(dataset, sample_weight_col, target_col, fractions)
    298 sample_weight_expr = create_map([lit(x) for x in chain(*fractions.items())])
    299 # rounded to 5 for better readability
    300 # e.g. fraction = 0.6 and the inverse is 1.6666667; (6 * 1.6666667) evaluates to 9.99999 instead of 10
--> 301 dataset = dataset.withColumn(
    302     sample_weight_col,
    303     round(col(sample_weight_col) * (1 / sample_weight_expr[col(target_col)]), 5))
    304 return dataset
File /databricks/spark/python/pyspark/instrumentation_utils.py:47, in _wrap_function.<locals>.wrapper(*args, **kwargs)
     45 start = time.perf_counter()
     46 try:
---> 47     res = func(*args, **kwargs)
     48     logger.log_success(
     49         module_name, class_name, function_name, time.perf_counter() - start, signature
     50     )
     51     return res
File /databricks/spark/python/pyspark/sql/dataframe.py:6282, in DataFrame.withColumn(self, colName, col)
   6277 if not isinstance(col, Column):
   6278     raise PySparkTypeError(
   6279         error_class="NOT_COLUMN",
   6280         message_parameters={"arg_name": "col", "arg_type": type(col).__name__},
   6281     )
-> 6282 return DataFrame(self._jdf.withColumn(colName, col._jc), self.sparkSession)
File /databricks/spark/python/lib/py4j-0.10.9.7-src.zip/py4j/java_gateway.py:1355, in JavaMember.__call__(self, *args)
   1349 command = proto.CALL_COMMAND_NAME +\
   1350     self.command_header +\
   1351     args_command +\
   1352     proto.END_COMMAND_PART
   1354 answer = self.gateway_client.send_command(command)
-> 1355 return_value = get_return_value(
   1356     answer, self.gateway_client, self.target_id, self.name)
   1358 for temp_arg in temp_args:
   1359     if hasattr(temp_arg, "_detach"):
File /databricks/spark/python/pyspark/errors/exceptions/captured.py:261, in capture_sql_exception.<locals>.deco(*a, **kw)
    257 converted = convert_exception(e.java_exception)
    258 if not isinstance(converted, UnknownException):
    259     # Hide where the exception came from that shows a non-Pythonic
    260     # JVM exception message.
--> 261     raise converted from None
    262 else:
    263     raise
```