In [3]:
```
import pandas as pd
```

In [ ]:

In [4]:
```
data=pd.read_csv('hr.csv')
```

In [ ]:

In [ ]:

In [5]:
```
data.head()
```

Out[5]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Education
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sci
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sci
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	(
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sci
4	27	No	Travel_Rarely	591	Research & Development	2	1	Me

5 rows × 35 columns

In [ ]:

In [6]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   Age                       1470 non-null   int64 
 1   Attrition                 1470 non-null   object
 2   BusinessTravel            1470 non-null   object
 3   DailyRate                 1470 non-null   int64 
 4   Department                1470 non-null   object
 5   DistanceFromHome          1470 non-null   int64 
 6   Education                 1470 non-null   int64 
 7   EducationField            1470 non-null   object
 8   EmployeeCount             1470 non-null   int64 
 9   EmployeeNumber            1470 non-null   int64 
 10  EnvironmentSatisfaction   1470 non-null   int64 
 11  Gender                    1470 non-null   object
 12  HourlyRate                1470 non-null   int64 
 13  JobInvolvement            1470 non-null   int64 
 14  JobLevel                  1470 non-null   int64 
 15  JobRole                   1470 non-null   object
 16  JobSatisfaction           1470 non-null   int64 
 17  MaritalStatus             1470 non-null   object
 18  MonthlyIncome             1470 non-null   int64 
 19  MonthlyRate               1470 non-null   int64 
 20  NumCompaniesWorked        1470 non-null   int64 
 21  Over18                    1470 non-null   object
 22  OverTime                  1470 non-null   object
 23  PercentSalaryHike         1470 non-null   int64 
 24  PerformanceRating         1470 non-null   int64 
 25  RelationshipSatisfaction  1470 non-null   int64 
 26  StandardHours             1470 non-null   int64 
 27  StockOptionLevel          1470 non-null   int64 
 28  TotalWorkingYears         1470 non-null   int64 
 29  TrainingTimesLastYear     1470 non-null   int64 
 30  WorkLifeBalance           1470 non-null   int64 
 31  YearsAtCompany            1470 non-null   int64 
 32  YearsInCurrentRole        1470 non-null   int64 
 33  YearsSinceLastPromotion   1470 non-null   int64 
 34  YearsWithCurrManager      1470 non-null   int64 
dtypes: int64(26), object(9)
memory usage: 402.1+ KB
```

In [ ]:

In [5]:
```
data.shape
```

Out[5]:
```
(1470, 35)
```

In [7]:
```
data.isnull().sum()
```

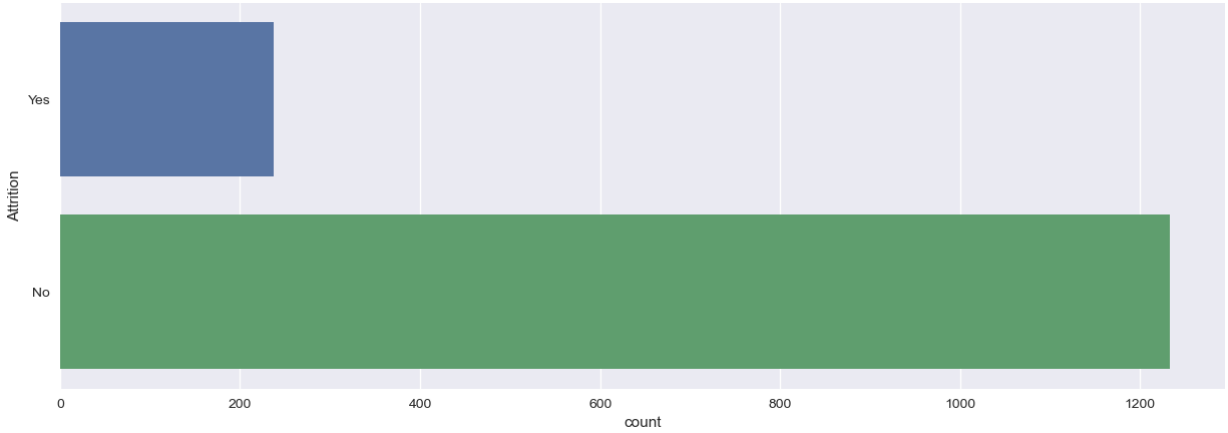
Out[7]:
```
Age                         0
Attrition                   0
BusinessTravel              0
DailyRate                   0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EmployeeCount               0
EmployeeNumber              0
EnvironmentSatisfaction     0
Gender                      0
HourlyRate                  0
JobInvolvement              0
JobLevel                    0
JobRole                     0
JobSatisfaction             0
MaritalStatus               0
MonthlyIncome               0
MonthlyRate                 0
NumCompaniesWorked          0
Over18                      0
OverTime                    0
PercentSalaryHike           0
PerformanceRating           0
RelationshipSatisfaction    0
StandardHours               0
StockOptionLevel            0
TotalWorkingYears           0
TrainingTimesLastYear       0
WorkLifeBalance             0
YearsAtCompany              0
YearsInCurrentRole          0
YearsSinceLastPromotion     0
YearsWithCurrManager        0
dtype: int64
```

In [31]:
```
print(data.duplicated().value_counts())
data.drop_duplicates(inplace=True)
print(len(data))
```

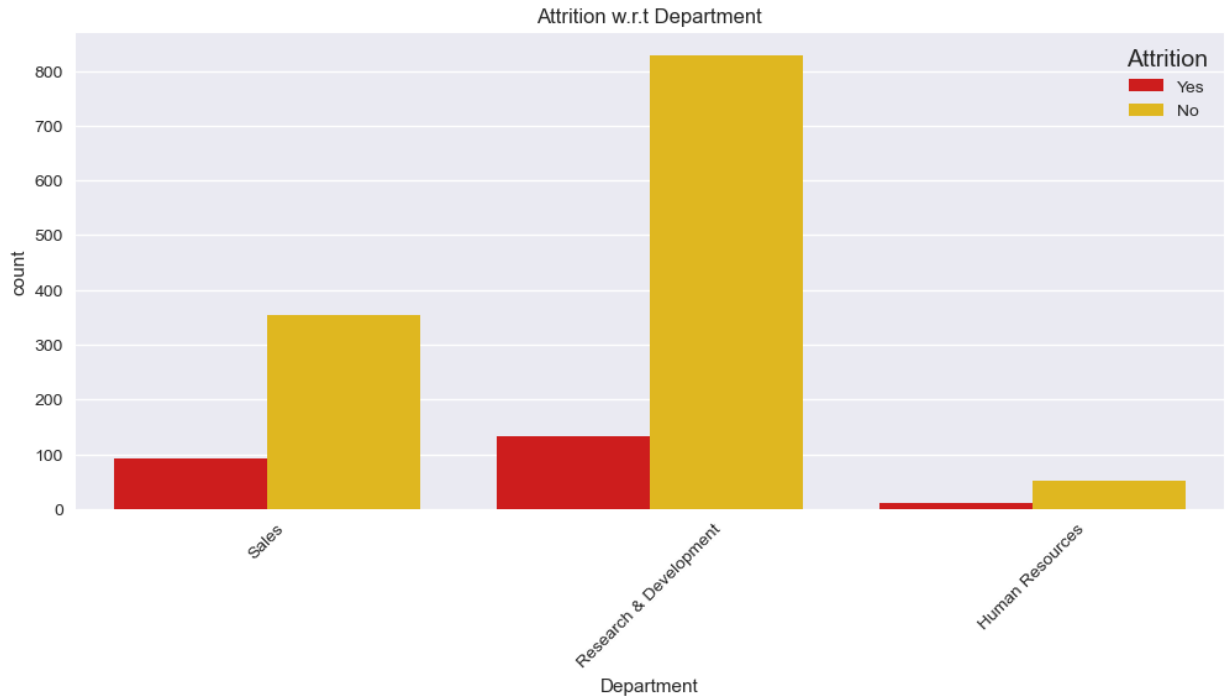
```
False    1470
dtype: int64
1470
```

In [33]:
```
plt.figure(figsize=(15,5))
plt.rc("font",size=14)
sns.countplot(y='Attrition',data=data)
```

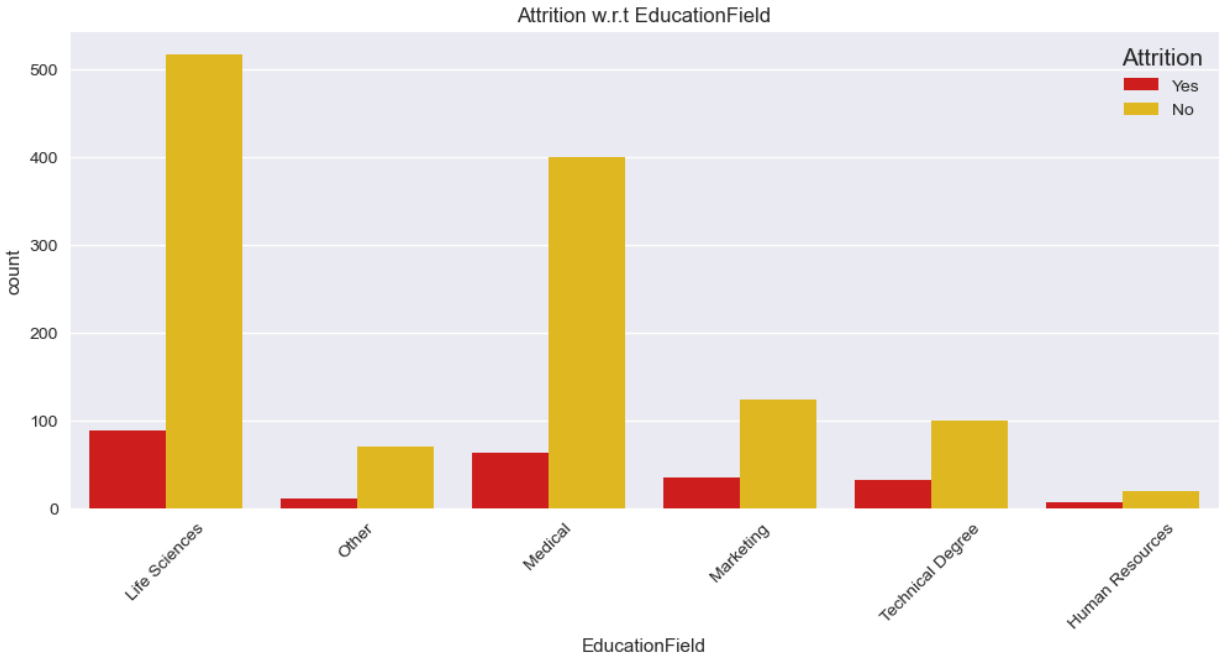
Out[33]:
```
<Axes: xlabel='count', ylabel='Attrition'>
```



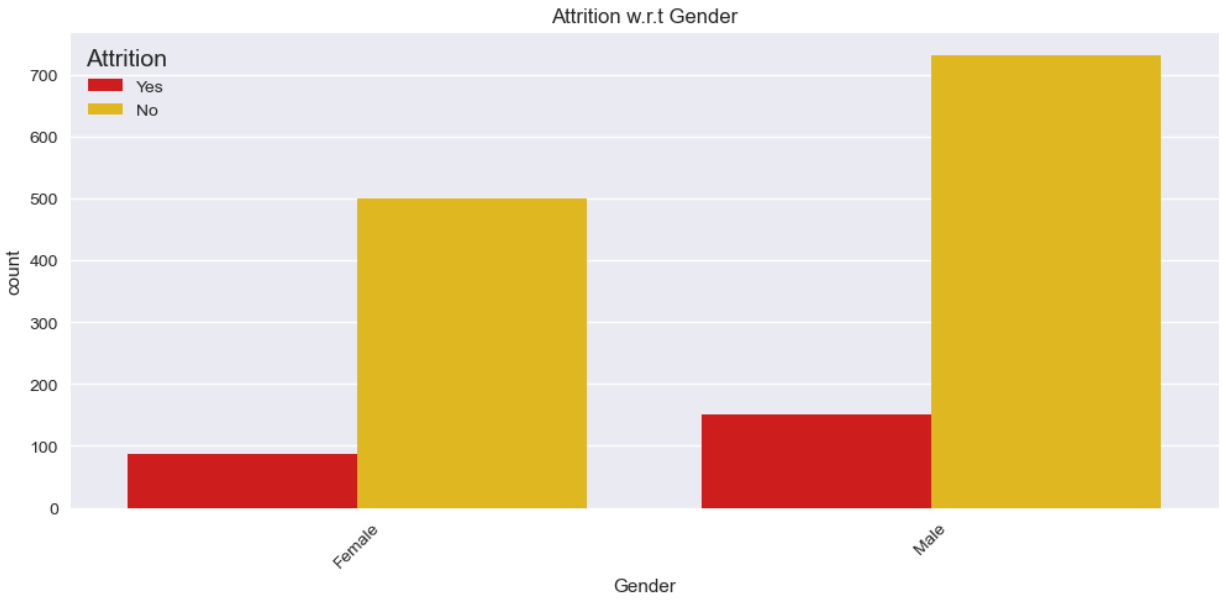
In [36]:
```
plt.figure(figsize=(12,5))
sns.countplot(x='Department',hue='Attrition', data=data, palette='hot')
plt.title("Attrition w.r.t Department")
plt.xticks(rotation=45)
plt.show()
```



In [37]:
```
plt.figure(figsize=(12,5))
sns.countplot(x='EducationField',hue='Attrition', data=data, palette='hot')
plt.title("Attrition w.r.t EducationField")
plt.xticks(rotation=45)
plt.show()
```



In [38]:
```
plt.figure(figsize=(12,5))
sns.countplot(x='Gender',hue='Attrition', data=data, palette='hot')
plt.title("Attrition w.r.t Gender")
plt.xticks(rotation=45)
plt.show()
```



In [39]:
```
data['Gender'].value_counts()
```

Out[39]:
```
Male      882
Female    588
Name: Gender, dtype: int64
```

In [40]:
```
data['count'] = 1
```

In [41]:
```
data.groupby(['Gender','Attrition']).agg({'count':'sum'})
```

Out[41]:

Gender	Attrition	count
Female	No	501
	Yes	87
Male	No	732
	Yes	150

In [42]:
```
87/(501+87)
```

Out[42]:
```
0.14795918367346939
```

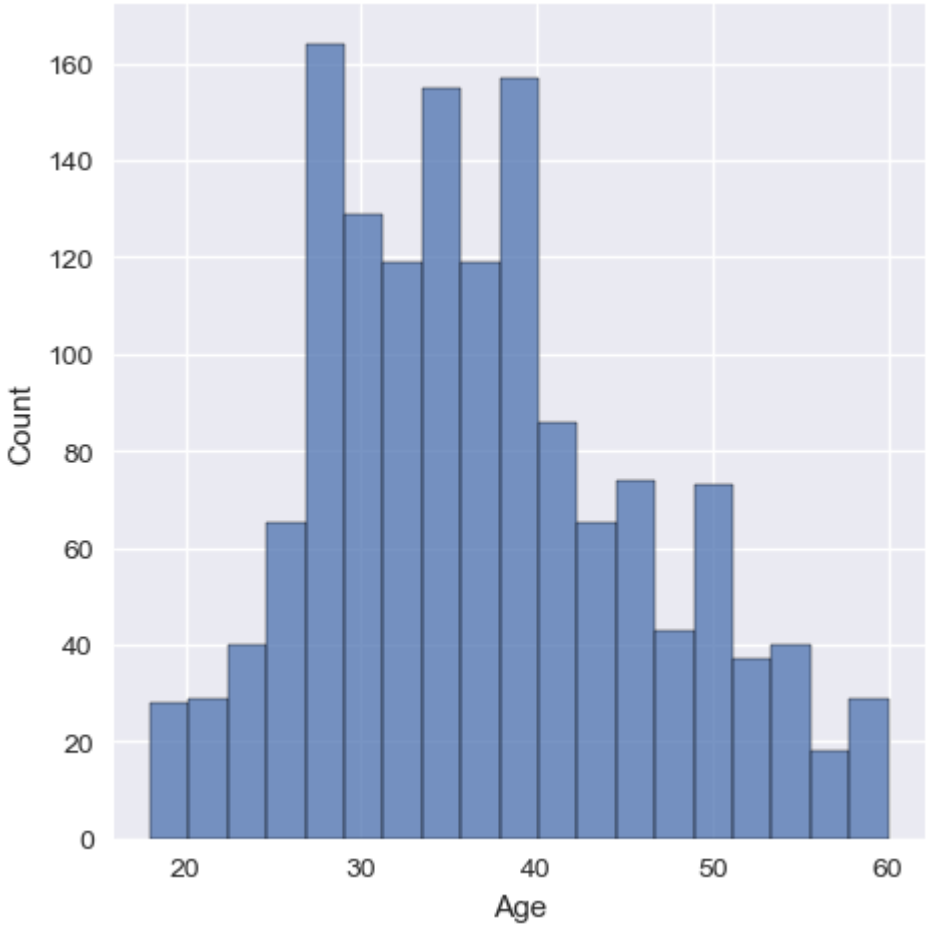
In [43]:
```
150/(735+50)
```

Out[43]:
```
0.1910828025477707
```

In [49]:
```
plt.figure(figsize=(12,5))
sns.displot(data['Age'])

plt.show()
```

```
<Figure size 1200x500 with 0 Axes>
```



In [57]:
```
from sklearn.preprocessing import LabelEncoder
encoding_cols=['BusinessTravel','Department','JobRole','MaritalStatus']
```

```
label_encoders = {}
for column in encoding_cols:
    label_encoders[column]= LabelEncoder()
    data[column] = label_encoders[column].fit_transform(data[column])
```

In [58]:
```
data.head()
```

Out[58]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationFi
0	41	Yes	2	1102	2	1	2	Life Scien
1	49	No	1	279	1	8	1	Life Scien
2	37	Yes	2	1373	1	2	2	Otl
3	33	No	1	1392	1	3	4	Life Scien
4	27	No	2	591	1	2	1	Medi

5 rows × 37 columns

In [27]:
```
X = data.drop('Attrition', axis=1)
y = data['Attrition']
```

In [ ]:
```

```

In [ ]:
```

```

In [18]:
```
from sklearn.model_selection import train_test_split
x_train  , x_test , y_train, y_test = train_test_split (X ,y ,test_size = 0.2 , random
```

In [21]:
```
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV
from sklearn.preprocessing import StandardScaler, OneHotEncoder , OrdinalEncoder
from sklearn.pipeline import make_pipeline
from sklearn.compose import ColumnTransformer
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler
```

In [22]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
```

In [23]:
```
X_train_under, y_train_under = RandomUnderSampler(random_state=42).fit_resample(X_trai
```

In [24]:
```
X_train_over, y_train_over = RandomOverSampler(random_state=42).fit_resample(X_train,
```

In [25]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
```

In [28]:
```
categorical_cols = ['Department', 'EducationField', 'Gender', 'JobRole', 'MaritalStatu
ordinal_cols = ['Education', 'EnvironmentSatisfaction', 'JobInvolvement', 'JobLevel',
                'PerformanceRating', 'RelationshipSatisfaction', 'StockOptionLevel', '
numerical_cols = ['Age','BusinessTravel', 'DistanceFromHome', 'HourlyRate', 'MonthlyIr
```

```
                  'PercentSalaryHike', 'TotalWorkingYears', 'TrainingTimesLastYear', '
                  'YearsInCurrentRole', 'YearsSinceLastPromotion', 'YearsWithCurrManag
```

In [31]:
```
categorical_transformer = make_pipeline(
    OneHotEncoder(drop='first', sparse=False)
)

ordinal_transformer = make_pipeline(
    OrdinalEncoder()
)

numerical_transformer = make_pipeline(
    StandardScaler()
)

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numerical_transformer, numerical_cols),
        ('cat', categorical_transformer, categorical_cols),
        ('ord', ordinal_transformer, ordinal_cols)
    ])


lr = make_pipeline(
    preprocessor,
    LogisticRegression(random_state=42)
)


rf = make_pipeline(
    preprocessor,
    RandomForestClassifier(random_state=42)
)


data = make_pipeline(
    preprocessor,
    DecisionTreeClassifier(random_state=42)
)

knn = make_pipeline(
    preprocessor,
    KNeighborsClassifier()
)
```

In [41]:
```
model = lr.fit(x_train, y_train)
y_pred = model.predict(x_test)
model.score(x_test, y_test)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[41], line 1
----> 1 model = lr.fit(x_train, y_train)
      2 y_pred = model.predict(x_test)
      3 model.score(x_test, y_test)

File ~\anaconda3\Lib\site-packages\sklearn\pipeline.py:401, in Pipeline.fit(self, X,
y, **fit_params)
    375 """Fit the model.
    376
    377 Fit all the transformers one after the other and transform the
   (...)
    398     Pipeline with fitted steps.
    399 """
    400 fit_params_steps = self._check_fit_params(**fit_params)
--> 401 Xt = self._fit(X, y, **fit_params_steps)
    402 with _print_elapsed_time("Pipeline", self._log_message(len(self.steps) - 1)):
    403     if self._final_estimator != "passthrough":

File ~\anaconda3\Lib\site-packages\sklearn\pipeline.py:359, in Pipeline._fit(self, X,
y, **fit_params_steps)
    357     cloned_transformer = clone(transformer)
    358 # Fit or load from cache the current transformer
--> 359 X, fitted_transformer = fit_transform_one_cached(
    360     cloned_transformer,
    361     X,
    362     y,
    363     None,
    364     message_clsname="Pipeline",
    365     message=self._log_message(step_idx),
    366     **fit_params_steps[name],
    367 )
    368 # Replace the transformer of the step with the fitted
    369 # transformer. This is necessary when loading the transformer
    370 # from the cache.
    371 self.steps[step_idx] = (name, fitted_transformer)

File ~\anaconda3\Lib\site-packages\joblib\memory.py:349, in NotMemorizedFunc.__call__
(self, *args, **kwargs)
    348 def __call__(self, *args, **kwargs):
--> 349     return self.func(*args, **kwargs)

File ~\anaconda3\Lib\site-packages\sklearn\pipeline.py:893, in _fit_transform_one(tra
nsformer, X, y, weight, message_clsname, message, **fit_params)
    891 with _print_elapsed_time(message_clsname, message):
    892     if hasattr(transformer, "fit_transform"):
--> 893         res = transformer.fit_transform(X, y, **fit_params)
    894     else:
    895         res = transformer.fit(X, y, **fit_params).transform(X)

File ~\anaconda3\Lib\site-packages\sklearn\utils\_set_output.py:140, in _wrap_method_
output.<locals>.wrapped(self, X, *args, **kwargs)
    138 @wraps(f)
    139 def wrapped(self, X, *args, **kwargs):
--> 140     data_to_wrap = f(self, X, *args, **kwargs)
    141     if isinstance(data_to_wrap, tuple):
    142         # only wrap the first output for cross decomposition
    143         return (
    144             _wrap_data_with_container(method, data_to_wrap[0], X, self),
```

```
    145             *data_to_wrap[1:],
    146         )

File ~\anaconda3\Lib\site-packages\sklearn\compose\_column_transformer.py:727, in ColumnTransformer.fit_transform(self, X, y)
    724 self._validate_column_callables(X)
    725 self._validate_remainder(X)
--> 727 result = self._fit_transform(X, y, _fit_transform_one)
    729 if not result:
    730     self._update_fitted_transformers([])

File ~\anaconda3\Lib\site-packages\sklearn\compose\_column_transformer.py:658, in ColumnTransformer._fit_transform(self, X, y, func, fitted, column_as_strings)
    652 transformers = list(
    653     self._iter(
    654         fitted=fitted, replace_strings=True, column_as_strings=column_as_strings
    655     )
    656 )
    657 try:
--> 658     return Parallel(n_jobs=self.n_jobs)(
    659         delayed(func)(
    660             transformer=clone(trans) if not fitted else trans,
    661             X=_safe_indexing(X, column, axis=1),
    662             y=y,
    663             weight=weight,
    664             message_clsname="ColumnTransformer",
    665             message=self._log_message(name, idx, len(transformers)),
    666         )
    667         for idx, (name, trans, column, weight) in enumerate(transformers, 1)
    668     )
    669 except ValueError as e:
    670     if "Expected 2D array, got 1D array instead" in str(e):

File ~\anaconda3\Lib\site-packages\sklearn\utils\parallel.py:63, in Parallel.__call__(self, iterable)
     58 config = get_config()
     59 iterable_with_config = (
     60     (_with_config(delayed_func, config), args, kwargs)
     61     for delayed_func, args, kwargs in iterable
     62 )
---> 63 return super().__call__(iterable_with_config)

File ~\anaconda3\Lib\site-packages\joblib\parallel.py:1048, in Parallel.__call__(self, iterable)
   1039 try:
   1040     # Only set self._iterating to True if at least a batch
   1041     # was dispatched. In particular this covers the edge
   (...)
   1045     # was very quick and its callback already dispatched all the
   1046     # remaining jobs.
   1047     self._iterating = False
-> 1048     if self.dispatch_one_batch(iterator):
   1049         self._iterating = self._original_iterator is not None
   1051     while self.dispatch_one_batch(iterator):

File ~\anaconda3\Lib\site-packages\joblib\parallel.py:864, in Parallel.dispatch_one_batch(self, iterator)
    862     return False
    863 else:
```

```
--> 864     self._dispatch(tasks)
    865     return True

File ~\anaconda3\Lib\site-packages\joblib\parallel.py:782, in Parallel._dispatch(sel
f, batch)
    780 with self._lock:
    781     job_idx = len(self._jobs)
--> 782     job = self._backend.apply_async(batch, callback=cb)
    783     # A job can complete so quickly than its callback is
    784     # called before we get here, causing self._jobs to
    785     # grow. To ensure correct results ordering, .insert is
    786     # used (rather than .append) in the following line
    787     self._jobs.insert(job_idx, job)

File ~\anaconda3\Lib\site-packages\joblib\_parallel_backends.py:208, in SequentialBac
kend.apply_async(self, func, callback)
    206 def apply_async(self, func, callback=None):
    207     """Schedule a func to be run"""
--> 208     result = ImmediateResult(func)
    209     if callback:
    210         callback(result)

File ~\anaconda3\Lib\site-packages\joblib\_parallel_backends.py:572, in ImmediateResu
lt.__init__(self, batch)
    569 def __init__(self, batch):
    570     # Don't delay the application, to avoid keeping the input
    571     # arguments in memory
--> 572     self.results = batch()

File ~\anaconda3\Lib\site-packages\joblib\parallel.py:263, in BatchedCalls.__call__(s
elf)
    259 def __call__(self):
    260     # Set the default nested backend to self._backend but do not set the
    261     # change the default number of processes to -1
    262     with parallel_backend(self._backend, n_jobs=self._n_jobs):
--> 263         return [func(*args, **kwargs)
    264                 for func, args, kwargs in self.items]

File ~\anaconda3\Lib\site-packages\joblib\parallel.py:263, in <listcomp>(.0)
    259 def __call__(self):
    260     # Set the default nested backend to self._backend but do not set the
    261     # change the default number of processes to -1
    262     with parallel_backend(self._backend, n_jobs=self._n_jobs):
--> 263         return [func(*args, **kwargs)
    264                 for func, args, kwargs in self.items]

File ~\anaconda3\Lib\site-packages\sklearn\utils\parallel.py:123, in _FuncWrapper.__c
all__(self, *args, **kwargs)
    121     config = {}
    122 with config_context(**config):
--> 123     return self.function(*args, **kwargs)

File ~\anaconda3\Lib\site-packages\sklearn\pipeline.py:893, in _fit_transform_one(tra
nsformer, X, y, weight, message_clsname, message, **fit_params)
    891 with _print_elapsed_time(message_clsname, message):
    892     if hasattr(transformer, "fit_transform"):
--> 893         res = transformer.fit_transform(X, y, **fit_params)
    894     else:
    895         res = transformer.fit(X, y, **fit_params).transform(X)
```

```
File ~\anaconda3\Lib\site-packages\sklearn\pipeline.py:445, in Pipeline.fit_transform
(self, X, y, **fit_params)
    443 fit_params_last_step = fit_params_steps[self.steps[-1][0]]
    444 if hasattr(last_step, "fit_transform"):
--> 445     return last_step.fit_transform(Xt, y, **fit_params_last_step)
    446 else:
    447     return last_step.fit(Xt, y, **fit_params_last_step).transform(Xt)

File ~\anaconda3\Lib\site-packages\sklearn\utils\_set_output.py:140, in _wrap_method_
output.<locals>.wrapped(self, X, *args, **kwargs)
    138 @wraps(f)
    139 def wrapped(self, X, *args, **kwargs):
--> 140     data_to_wrap = f(self, X, *args, **kwargs)
    141     if isinstance(data_to_wrap, tuple):
    142         # only wrap the first output for cross decomposition
    143         return (
    144             _wrap_data_with_container(method, data_to_wrap[0], X, self),
    145             *data_to_wrap[1:],
    146         )

File ~\anaconda3\Lib\site-packages\sklearn\base.py:881, in TransformerMixin.fit_trans
form(self, X, y, **fit_params)
    878     return self.fit(X, **fit_params).transform(X)
    879 else:
    880     # fit method of arity 2 (supervised transformation)
--> 881     return self.fit(X, y, **fit_params).transform(X)

File ~\anaconda3\Lib\site-packages\sklearn\preprocessing\_data.py:824, in StandardSca
ler.fit(self, X, y, sample_weight)
    822 # Reset internal state before fitting
    823 self._reset()
--> 824 return self.partial_fit(X, y, sample_weight)

File ~\anaconda3\Lib\site-packages\sklearn\preprocessing\_data.py:861, in StandardSca
ler.partial_fit(self, X, y, sample_weight)
    858 self._validate_params()
    860 first_call = not hasattr(self, "n_samples_seen_")
--> 861 X = self._validate_data(
    862     X,
    863     accept_sparse=("csr", "csc"),
    864     dtype=FLOAT_DTYPES,
    865     force_all_finite="allow-nan",
    866     reset=first_call,
    867 )
    868 n_features = X.shape[1]
    870 if sample_weight is not None:

File ~\anaconda3\Lib\site-packages\sklearn\base.py:565, in BaseEstimator._validate_da
ta(self, X, y, reset, validate_separately, **check_params)
    563     raise ValueError("Validation should be done on X, y or both.")
    564 elif not no_val_X and no_val_y:
--> 565     X = check_array(X, input_name="X", **check_params)
    566     out = X
    567 elif no_val_X and not no_val_y:

File ~\anaconda3\Lib\site-packages\sklearn\utils\validation.py:879, in check_array(ar
ray, accept_sparse, accept_large_sparse, dtype, order, copy, force_all_finite, ensure
_2d, allow_nd, ensure_min_samples, ensure_min_features, estimator, input_name)
    877         array = xp.astype(array, dtype, copy=False)
    878     else:
```

```
--> 879         array = _asarray_with_order(array, order=order, dtype=dtype, xp=xp)
    880 except ComplexWarning as complex_warning:
    881     raise ValueError(
    882         "Complex data not supported\n{}\n".format(array)
    883     ) from complex_warning

File ~\anaconda3\Lib\site-packages\sklearn\utils\_array_api.py:185, in _asarray_with_
order(array, dtype, order, copy, xp)
    182     xp, _ = get_namespace(array)
    183 if xp.__name__ in {"numpy", "numpy.array_api"}:
    184     # Use NumPy API to support order
--> 185     array = numpy.asarray(array, order=order, dtype=dtype)
    186     return xp.asarray(array, copy=copy)
    187 else:

File ~\anaconda3\Lib\site-packages\pandas\core\generic.py:2070, in NDFrame.__array__
(self, dtype)
   2069 def __array__(self, dtype: npt.DTypeLike | None = None) -> np.ndarray:
-> 2070     return np.asarray(self._values, dtype=dtype)

ValueError: could not convert string to float: 'Travel_Rarely'
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: