Here

|  |  |
| --- | --- |
| First Attempt | Understand the following scenario and memorize it because I will many questions from this:  IEEE-CIS Fraud Detection  Can you detect fraud from customer transactions?  Description  Imagine standing at the check-out counter at the grocery store with a long line behind you and the cashier not-so-quietly announces that your card has been declined. In this moment, you probably aren’t thinking about the data science that determined your fate.  Embarrassed, and certain you have the funds to cover everything needed for an epic nacho party for 50 of your closest friends, you try your card again. Same result. As you step aside and allow the cashier to tend to the next customer, you receive a text message from your bank. “Press 1 if you really tried to spend $500 on cheddar cheese.”  While perhaps cumbersome (and often embarrassing) in the moment, this fraud prevention system is actually saving consumers millions of dollars per year. Researchers from the IEEE Computational Intelligence Society (IEEE-CIS) want to improve this figure, while also improving the customer experience. With higher accuracy fraud detection, you can get on with your chips without the hassle.  IEEE-CIS works across a variety of AI and machine learning areas, including deep neural networks, fuzzy systems, evolutionary computation, and swarm intelligence. Today they’re partnering with the world’s leading payment service company, Vesta Corporation, seeking the best solutions for fraud prevention industry, and now you are invited to join the challenge.  In this competition, you’ll benchmark machine learning models on a challenging large-scale dataset. The data comes from Vesta's real-world e-commerce transactions and contains a wide range of features from device type to product features. You also have the opportunity to create new features to improve your results.  If successful, you’ll improve the efficacy of fraudulent transaction alerts for millions of people around the world, helping hundreds of thousands of businesses reduce their fraud loss and increase their revenue. And of course, you will save party people just like you the hassle of false positives.  Vesta Corporation provided the dataset for this competition. Vesta Corporation is the forerunner in guaranteed e-commerce payment solutions. Founded in 1995, Vesta pioneered the process of fully guaranteed card-not-present (CNP) payment transactions for the telecommunications industry. Since then, Vesta has firmly expanded data science and machine learning capabilities across the globe and solidified its position as the leader in guaranteed ecommerce payments. Today, Vesta guarantees more than $18B in transactions annually.  Evaluation  Submissions are evaluated on area under the ROC curve between the predicted probability and the observed target.  We have Four type of CSV files, train identity, train transaction, test identity and test transaction. Let explain the column of each files one by one.  Train Identity has the following column:  TransactionID id\_01 id\_02 id\_03 id\_04 id\_05 id\_06 id\_07 id\_08 id\_09 id\_10 id\_11 id\_12 id\_13 id\_14 id\_15 id\_16 id\_17 id\_18 id\_19 id\_20 id\_21 id\_22 id\_23 id\_24 id\_25 id\_26 id\_27 id\_28 id\_29 id\_30 id\_31 id\_32 id\_33 id\_34 id\_35 id\_36 id\_37 id\_38 DeviceType DeviceInfo  2987004 0 70787 100 NotFound -480 New NotFound 166 542 144 New NotFound Android 7.0 samsung browser 6.2 32 2220x1080 match\_status:2 T F T T mobile SAMSUNG SM-G892A Build/NRD90M  2987008 -5 98945 0 -5 100 NotFound 49 -300 New NotFound 166 621 500 New NotFound iOS 11.1.2 mobile safari 11.0 32 1334x750 match\_status:1 T F F T mobile iOS Device  2987010 -5 191631 0 0 0 0 0 0 100 NotFound 52 Found Found 121 410 142 Found Found chrome 62.0 F F T T desktop Windows  2987011 -5 221832 0 -6 100 NotFound 52 New NotFound 225 176 507 New NotFound chrome 62.0 F F T T desktop  The train transaction file has the following column:  TransactionID isFraud TransactionDT TransactionAmt ProductCD card1 card2 card3 card4 card5 card6 addr1 addr2 dist1 dist2 P\_emaildomain R\_emaildomain C1 C2 C3 C4 C5 C6 C7 C8 C9 C10 C11 C12 C13 C14 D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12 D13 D14 D15 M1 M2 M3 M4 M5 M6 M7 M8 M9 V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20 V21 V22 V23 V24 V25 V26 V27 V28 V29 V30 V31 V32 V33 V34 V35 V36 V37 V38 V39 V40 V41 V42 V43 V44 V45 V46 V47 V48 V49 V50 V51 V52 V53 V54 V55 V56 V57 V58 V59 V60 V61 V62 V63 V64 V65 V66 V67 V68 V69 V70 V71 V72 V73 V74 V75 V76 V77 V78 V79 V80 V81 V82 V83 V84 V85 V86 V87 V88 V89 V90 V91 V92 V93 V94 V95 V96 V97 V98 V99 V100 V101 V102 V103 V104 V105 V106 V107 V108 V109 V110 V111 V112 V113 V114 V115 V116 V117 V118 V119 V120 V121 V122 V123 V124 V125 V126 V127 V128 V129 V130 V131 V132 V133 V134 V135 V136 V137 V138 V139 V140 V141 V142 V143 V144 V145 V146 V147 V148 V149 V150 V151 V152 V153 V154 V155 V156 V157 V158 V159 V160 V161 V162 V163 V164 V165 V166 V167 V168 V169 V170 V171 V172 V173 V174 V175 V176 V177 V178 V179 V180 V181 V182 V183 V184 V185 V186 V187 V188 V189 V190 V191 V192 V193 V194 V195 V196 V197 V198 V199 V200 V201 V202 V203 V204 V205 V206 V207 V208 V209 V210 V211 V212 V213 V214 V215 V216 V217 V218 V219 V220 V221 V222 V223 V224 V225 V226 V227 V228 V229 V230 V231 V232 V233 V234 V235 V236 V237 V238 V239 V240 V241 V242 V243 V244 V245 V246 V247 V248 V249 V250 V251 V252 V253 V254 V255 V256 V257 V258 V259 V260 V261 V262 V263 V264 V265 V266 V267 V268 V269 V270 V271 V272 V273 V274 V275 V276 V277 V278 V279 V280 V281 V282 V283 V284 V285 V286 V287 V288 V289 V290 V291 V292 V293 V294 V295 V296 V297 V298 V299 V300 V301 V302 V303 V304 V305 V306 V307 V308 V309 V310 V311 V312 V313 V314 V315 V316 V317 V318 V319 V320 V321 V322 V323 V324 V325 V326 V327 V328 V329 V330 V331 V332 V333 V334 V335 V336 V337 V338 V339  2987000 0 86400 68.5 W 13926 150 discover 142 credit 315 87 19 1 1 0 0 0 1 0 0 1 0 2 0 1 1 14 13 13 13 0 T T T M2 F T 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0 0 0 0 1 1 0 0 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 1 0 0 1 1 1 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 117 0 0 0 0 0 117 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 1 0 117 0 0 0 0 0 0 0 0 0 117 0 0 0 0  The target variable isFraud. The test identity and test transaction has the same column except isFraud. they don't have isFraud column and we have to predict the isFraud. |
| Second Attempt | Now train the best model and predict isFraud in test identity and test transaction |
| Third Attempt | I got the following error while training with the following code:  # Initialize and train the model  model = RandomForestClassifier(n\_estimators=100, random\_state=42)  model.fit(X\_train, y\_train)  # Validate the model  y\_val\_pred = model.predict\_proba(X\_val)[:, 1]  auc\_score = roc\_auc\_score(y\_val, y\_val\_pred)  print(f'Validation AUC Score: {auc\_score}')  ---------------------------------------------------------------------------  ValueError Traceback (most recent call last)  ~\AppData\Local\Temp\ipykernel\_25712\2249396114.py in ?()  ----> 3 # Initialize and train the model  4 model = RandomForestClassifier(n\_estimators=100, random\_state=42)  5 model.fit(X\_train, y\_train)  6  ~\Anaconda3\envs\MLPR\lib\site-packages\sklearn\base.py in ?(estimator, \*args, \*\*kwargs)  1148 skip\_parameter\_validation=(  1149 prefer\_skip\_nested\_validation or global\_skip\_validation  1150 )  1151 ):  -> 1152 return fit\_method(estimator, \*args, \*\*kwargs)  ~\Anaconda3\envs\MLPR\lib\site-packages\sklearn\ensemble\\_forest.py in ?(self, X, y, sample\_weight)  344 """  345 # Validate or convert input data  346 if issparse(y):  347 raise ValueError("sparse multilabel-indicator for y is not supported.")  --> 348 X, y = self.\_validate\_data(  349 X, y, multi\_output=True, accept\_sparse="csc", dtype=DTYPE  350 )  351 if sample\_weight is not None:  ~\Anaconda3\envs\MLPR\lib\site-packages\sklearn\base.py in ?(self, X, y, reset, validate\_separately, cast\_to\_ndarray, \*\*check\_params)  618 if "estimator" not in check\_y\_params:  619 check\_y\_params = {\*\*default\_check\_params, \*\*check\_y\_params}  620 y = check\_array(y, input\_name="y", \*\*check\_y\_params)  621 else:  --> 622 X, y = check\_X\_y(X, y, \*\*check\_params)  623 out = X, y  624  625 if not no\_val\_X and check\_params.get("ensure\_2d", True):  ~\Anaconda3\envs\MLPR\lib\site-packages\sklearn\utils\validation.py in ?(X, y, accept\_sparse, accept\_large\_sparse, dtype, order, copy, force\_all\_finite, ensure\_2d, allow\_nd, multi\_output, ensure\_min\_samples, ensure\_min\_features, y\_numeric, estimator)  1142 raise ValueError(  1143 f"{estimator\_name} requires y to be passed, but the target y is None"  1144 )  1145  -> 1146 X = check\_array(  1147 X,  1148 accept\_sparse=accept\_sparse,  1149 accept\_large\_sparse=accept\_large\_sparse,  ~\Anaconda3\envs\MLPR\lib\site-packages\sklearn\utils\validation.py in ?(array, accept\_sparse, accept\_large\_sparse, dtype, order, copy, force\_all\_finite, ensure\_2d, allow\_nd, ensure\_min\_samples, ensure\_min\_features, estimator, input\_name)  913 array = xp.astype(array, dtype, copy=False)  914 else:  915 array = \_asarray\_with\_order(array, order=order, dtype=dtype, xp=xp)  916 except ComplexWarning as complex\_warning:  --> 917 raise ValueError(  918 "Complex data not supported\n{}\n".format(array)  919 ) from complex\_warning  920  ~\Anaconda3\envs\MLPR\lib\site-packages\sklearn\utils\\_array\_api.py in ?(array, dtype, order, copy, xp)  376 # Use NumPy API to support order  377 if copy is True:  378 array = numpy.array(array, order=order, dtype=dtype)  379 else:  --> 380 array = numpy.asarray(array, order=order, dtype=dtype)  381  382 # At this point array is a NumPy ndarray. We convert it to an array  383 # container that is consistent with the input's namespace.  ~\Anaconda3\envs\MLPR\lib\site-packages\pandas\core\generic.py in ?(self, dtype)  1996 def \_\_array\_\_(self, dtype: npt.DTypeLike | None = None) -> np.ndarray:  1997 values = self.\_values  -> 1998 arr = np.asarray(values, dtype=dtype)  1999 if (  2000 astype\_is\_view(values.dtype, arr.dtype)  2001 and using\_copy\_on\_write()  ValueError: could not convert string to float: 'W'  The particular datatype is give below.  # Fill missing values  train\_data.fillna(-999, inplace=True)  test\_data.fillna(-999, inplace=True)  # Convert categorical features to numerical  categorical\_features = ['DeviceType', 'DeviceInfo'] # Add more categorical features if needed  for feature in categorical\_features:  train\_data[feature] = train\_data[feature].astype('category').cat.codes  test\_data[feature] = test\_data[feature].astype('category').cat.codes  # Define features and target variable  X = train\_data.drop(columns=['TransactionID', 'isFraud'])  y = train\_data['isFraud']  X\_test = test\_data.drop(columns=['TransactionID'])  TransactionID int64  id\_01 float64  id\_02 float64  id\_03 float64  id\_04 float64  id\_05 float64  id\_06 float64  id\_07 float64  id\_08 float64  id\_09 float64  id\_10 float64  id\_11 float64  id\_12 object  id\_13 float64  id\_14 float64  id\_15 object  id\_16 object  id\_17 float64  id\_18 float64  id\_19 float64  id\_20 float64  id\_21 float64  id\_22 float64  id\_23 object  id\_24 float64  id\_25 float64  id\_26 float64  id\_27 object  id\_28 object  id\_29 object  id\_30 object  id\_31 object  id\_32 float64  id\_33 object  id\_34 object  id\_35 object  id\_36 object  id\_37 object  id\_38 object  DeviceType object  DeviceInfo object  dtype: object  TransactionID int64  isFraud int64  TransactionDT int64  TransactionAmt float64  ProductCD object  ...  V335 float64  V336 float64  V337 float64  V338 float64  V339 float64  Length: 394, dtype: object |
| Fourth Attempt | # Identify columns with non-numeric data  non\_numeric\_columns = X.select\_dtypes(include=['object']).columns  print(f"Non-numeric columns: {non\_numeric\_columns}")  # Convert categorical features to numeric  for feature in non\_numeric\_columns:  X[feature] = X[feature].astype('category').cat.codes  X\_test[feature] = X\_test[feature].astype('category').cat.codes  # Fill remaining missing values  X.fillna(-999, inplace=True)  X\_test.fillna(-999, inplace=True)  # Ensure that all data is numeric  print(X.dtypes)  print(X\_test.dtypes)  ---------------------------------------------------------------------------  KeyError Traceback (most recent call last)  File ~\Anaconda3\envs\MLPR\lib\site-packages\pandas\core\indexes\base.py:3653, in Index.get\_loc(self, key)  3652 try:  -> 3653 return self.\_engine.get\_loc(casted\_key)  3654 except KeyError as err:  File ~\Anaconda3\envs\MLPR\lib\site-packages\pandas\\_libs\index.pyx:147, in pandas.\_libs.index.IndexEngine.get\_loc()  File ~\Anaconda3\envs\MLPR\lib\site-packages\pandas\\_libs\index.pyx:176, in pandas.\_libs.index.IndexEngine.get\_loc()  File pandas\\_libs\hashtable\_class\_helper.pxi:7080, in pandas.\_libs.hashtable.PyObjectHashTable.get\_item()  File pandas\\_libs\hashtable\_class\_helper.pxi:7088, in pandas.\_libs.hashtable.PyObjectHashTable.get\_item()  KeyError: 'id\_12'  The above exception was the direct cause of the following exception:  KeyError Traceback (most recent call last)  Cell In[11], line 8  6 for feature in non\_numeric\_columns:  7 X[feature] = X[feature].astype('category').cat.codes  ----> 8 X\_test[feature] = X\_test[feature].astype('category').cat.codes  10 # Fill remaining missing values  11 X.fillna(-999, inplace=True)  File ~\Anaconda3\envs\MLPR\lib\site-packages\pandas\core\frame.py:3761, in DataFrame.\_\_getitem\_\_(self, key)  3759 if self.columns.nlevels > 1:  3760 return self.\_getitem\_multilevel(key)  -> 3761 indexer = self.columns.get\_loc(key)  3762 if is\_integer(indexer):  3763 indexer = [indexer]  File ~\Anaconda3\envs\MLPR\lib\site-packages\pandas\core\indexes\base.py:3655, in Index.get\_loc(self, key)  3653 return self.\_engine.get\_loc(casted\_key)  3654 except KeyError as err:  -> 3655 raise KeyError(key) from err  3656 except TypeError:  3657 # If we have a listlike key, \_check\_indexing\_error will raise  3658 # InvalidIndexError. Otherwise we fall through and re-raise  3659 # the TypeError.  3660 self.\_check\_indexing\_error(key)  KeyError: 'id\_12' |
| Fifth Attempt | from sklearn.ensemble import RandomForestClassifier  from sklearn.metrics import roc\_auc\_score  from sklearn.model\_selection import train\_test\_split  # Define features and target variable  X = train\_data.drop(columns=['TransactionID', 'isFraud'])  y = train\_data['isFraud']  X\_test = test\_data.drop(columns=['TransactionID'])  # Split the training data into training and validation sets  X\_train, X\_val, y\_train, y\_val = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  # Initialize and train the model  model = RandomForestClassifier(n\_estimators=100, random\_state=42)  model.fit(X\_train, y\_train)  # Validate the model  y\_val\_pred = model.predict\_proba(X\_val)[:, 1]  auc\_score = roc\_auc\_score(y\_val, y\_val\_pred)  print(f'Validation AUC Score: {auc\_score}')  # Predict on the test set  test\_predictions = model.predict\_proba(X\_test)[:, 1]  # Prepare submission DataFrame  submission = pd.DataFrame({  'TransactionID': test\_data['TransactionID'],  'isFraud': test\_predictions  })  # Save to CSV  submission.to\_csv('submission.csv', index=False)  ---------------------------------------------------------------------------  ValueError Traceback (most recent call last)  ~\AppData\Local\Temp\ipykernel\_25712\180598639.py in ?()  ---> 15 from sklearn.ensemble import RandomForestClassifier  16 from sklearn.metrics import roc\_auc\_score  17 from sklearn.model\_selection import train\_test\_split  18  ~\Anaconda3\envs\MLPR\lib\site-packages\sklearn\base.py in ?(estimator, \*args, \*\*kwargs)  1148 skip\_parameter\_validation=(  1149 prefer\_skip\_nested\_validation or global\_skip\_validation  1150 )  1151 ):  -> 1152 return fit\_method(estimator, \*args, \*\*kwargs)  ~\Anaconda3\envs\MLPR\lib\site-packages\sklearn\ensemble\\_forest.py in ?(self, X, y, sample\_weight)  344 """  345 # Validate or convert input data  346 if issparse(y):  347 raise ValueError("sparse multilabel-indicator for y is not supported.")  --> 348 X, y = self.\_validate\_data(  349 X, y, multi\_output=True, accept\_sparse="csc", dtype=DTYPE  350 )  351 if sample\_weight is not None:  ~\Anaconda3\envs\MLPR\lib\site-packages\sklearn\base.py in ?(self, X, y, reset, validate\_separately, cast\_to\_ndarray, \*\*check\_params)  618 if "estimator" not in check\_y\_params:  619 check\_y\_params = {\*\*default\_check\_params, \*\*check\_y\_params}  620 y = check\_array(y, input\_name="y", \*\*check\_y\_params)  621 else:  --> 622 X, y = check\_X\_y(X, y, \*\*check\_params)  623 out = X, y  624  625 if not no\_val\_X and check\_params.get("ensure\_2d", True):  ~\Anaconda3\envs\MLPR\lib\site-packages\sklearn\utils\validation.py in ?(X, y, accept\_sparse, accept\_large\_sparse, dtype, order, copy, force\_all\_finite, ensure\_2d, allow\_nd, multi\_output, ensure\_min\_samples, ensure\_min\_features, y\_numeric, estimator)  1142 raise ValueError(  1143 f"{estimator\_name} requires y to be passed, but the target y is None"  1144 )  1145  -> 1146 X = check\_array(  1147 X,  1148 accept\_sparse=accept\_sparse,  1149 accept\_large\_sparse=accept\_large\_sparse,  ~\Anaconda3\envs\MLPR\lib\site-packages\sklearn\utils\validation.py in ?(array, accept\_sparse, accept\_large\_sparse, dtype, order, copy, force\_all\_finite, ensure\_2d, allow\_nd, ensure\_min\_samples, ensure\_min\_features, estimator, input\_name)  913 array = xp.astype(array, dtype, copy=False)  914 else:  915 array = \_asarray\_with\_order(array, order=order, dtype=dtype, xp=xp)  916 except ComplexWarning as complex\_warning:  --> 917 raise ValueError(  918 "Complex data not supported\n{}\n".format(array)  919 ) from complex\_warning  920  ~\Anaconda3\envs\MLPR\lib\site-packages\sklearn\utils\\_array\_api.py in ?(array, dtype, order, copy, xp)  376 # Use NumPy API to support order  377 if copy is True:  378 array = numpy.array(array, order=order, dtype=dtype)  379 else:  --> 380 array = numpy.asarray(array, order=order, dtype=dtype)  381  382 # At this point array is a NumPy ndarray. We convert it to an array  383 # container that is consistent with the input's namespace.  ~\Anaconda3\envs\MLPR\lib\site-packages\pandas\core\generic.py in ?(self, dtype)  1996 def \_\_array\_\_(self, dtype: npt.DTypeLike | None = None) -> np.ndarray:  1997 values = self.\_values  -> 1998 arr = np.asarray(values, dtype=dtype)  1999 if (  2000 astype\_is\_view(values.dtype, arr.dtype)  2001 and using\_copy\_on\_write()  ValueError: could not convert string to float: 'W' |
| Sixth Attempt | # Identify columns with non-numeric data  non\_numeric\_columns\_X = X.select\_dtypes(include=['object']).columns  non\_numeric\_columns\_X\_test = X\_test.select\_dtypes(include=['object']).columns  print(f"Non-numeric columns in X: {non\_numeric\_columns\_X}")  print(f"Non-numeric columns in X\_test: {non\_numeric\_columns\_X\_test}")  Non-numeric columns in X: Index(['ProductCD', 'card4', 'card6', 'P\_emaildomain', 'R\_emaildomain', 'M1',  'M2', 'M3', 'M4', 'M5', 'M6', 'M7', 'M8', 'M9', 'id\_12', 'id\_15',  'id\_16', 'id\_23', 'id\_27', 'id\_28', 'id\_29', 'id\_30', 'id\_31', 'id\_33',  'id\_34', 'id\_35', 'id\_36', 'id\_37', 'id\_38'],  dtype='object')  Non-numeric columns in X\_test: Index(['ProductCD', 'card4', 'card6', 'P\_emaildomain', 'R\_emaildomain', 'M1',  'M2', 'M3', 'M4', 'M5', 'M6', 'M7', 'M8', 'M9', 'id-12', 'id-15',  'id-16', 'id-23', 'id-27', 'id-28', 'id-29', 'id-30', 'id-31', 'id-33',  'id-34', 'id-35', 'id-36', 'id-37', 'id-38'],  dtype='object') |
| 7th Attempt | Validation AUC Score: 0.9346783032634545  ---------------------------------------------------------------------------  ValueError Traceback (most recent call last)  Cell In[18], line 18  15 print(f'Validation AUC Score: {auc\_score}')  17 # Predict on the test set  ---> 18 test\_predictions = model.predict\_proba(X\_test)[:, 1]  20 # Prepare submission DataFrame  21 submission = pd.DataFrame({  22 'TransactionID': test\_data['TransactionID'],  23 'isFraud': test\_predictions  24 })  File ~\Anaconda3\envs\MLPR\lib\site-packages\sklearn\ensemble\\_forest.py:865, in ForestClassifier.predict\_proba(self, X)  863 check\_is\_fitted(self)  864 # Check data  --> 865 X = self.\_validate\_X\_predict(X)  867 # Assign chunk of trees to jobs  868 n\_jobs, \_, \_ = \_partition\_estimators(self.n\_estimators, self.n\_jobs)  File ~\Anaconda3\envs\MLPR\lib\site-packages\sklearn\ensemble\\_forest.py:599, in BaseForest.\_validate\_X\_predict(self, X)  596 """  597 Validate X whenever one tries to predict, apply, predict\_proba."""  598 check\_is\_fitted(self)  --> 599 X = self.\_validate\_data(X, dtype=DTYPE, accept\_sparse="csr", reset=False)  600 if issparse(X) and (X.indices.dtype != np.intc or X.indptr.dtype != np.intc):  601 raise ValueError("No support for np.int64 index based sparse matrices")  File ~\Anaconda3\envs\MLPR\lib\site-packages\sklearn\base.py:580, in BaseEstimator.\_validate\_data(self, X, y, reset, validate\_separately, cast\_to\_ndarray, \*\*check\_params)  509 def \_validate\_data(  510 self,  511 X="no\_validation",  (...)  516 \*\*check\_params,  517 ):  518 """Validate input data and set or check the n\_features\_in\_ attribute.  519  520 Parameters  (...)  578 validated.  579 """  --> 580 self.\_check\_feature\_names(X, reset=reset)  582 if y is None and self.\_get\_tags()["requires\_y"]:  583 raise ValueError(  584 f"This {self.\_\_class\_\_.\_\_name\_\_} estimator "  585 "requires y to be passed, but the target y is None."  586 )  File ~\Anaconda3\envs\MLPR\lib\site-packages\sklearn\base.py:507, in BaseEstimator.\_check\_feature\_names(self, X, reset)  502 if not missing\_names and not unexpected\_names:  503 message += (  504 "Feature names must be in the same order as they were in fit.\n"  505 )  --> 507 raise ValueError(message)  ValueError: The feature names should match those that were passed during fit.  Feature names unseen at fit time:  - id-01  - id-02  - id-03  - id-04  - id-05  - ...  Feature names seen at fit time, yet now missing:  - id\_01  - id\_02  - id\_03  - id\_04  - id\_05  - ... |
| 8th Attempt | # Ensure consistency in column names  X.columns = X.columns.str.replace('-', '\_', regex=False)  X\_test.columns = X\_test.columns.str.replace('-', '\_', regex=False)  # Re-check non-numeric columns after renaming  non\_numeric\_columns\_X = X.select\_dtypes(include=['object']).columns  non\_numeric\_columns\_X\_test = X\_test.select\_dtypes(include=['object']).columns  print(f"Non-numeric columns in X: {non\_numeric\_columns\_X}")  print(f"Non-numeric columns in X\_test: {non\_numeric\_columns\_X\_test}")  Non-numeric columns in X: Index([], dtype='object')  Non-numeric columns in X\_test: Index([], dtype='object') |
| 9th Attempt | # Check if columns in X\_test match those in X  if X.columns.equals(X\_test.columns):  print("Columns in X and X\_test match.")  else:  missing\_cols = [col for col in X.columns if col not in X\_test.columns]  extra\_cols = [col for col in X\_test.columns if col not in X.columns]  print(f"Missing columns in X\_test: {missing\_cols}")  print(f"Extra columns in X\_test: {extra\_cols}")  result: Columns in X and X\_test match. |
| 10th Attempt | Columns in X and X\_test match.  ---------------------------------------------------------------------------  ValueError Traceback (most recent call last)  ~\AppData\Local\Temp\ipykernel\_25712\3689824276.py in ?()  ---> 33 from sklearn.ensemble import RandomForestClassifier  34 from sklearn.metrics import roc\_auc\_score  35 from sklearn.model\_selection import train\_test\_split  36  ~\Anaconda3\envs\MLPR\lib\site-packages\sklearn\base.py in ?(estimator, \*args, \*\*kwargs)  1148 skip\_parameter\_validation=(  1149 prefer\_skip\_nested\_validation or global\_skip\_validation  1150 )  1151 ):  -> 1152 return fit\_method(estimator, \*args, \*\*kwargs)  ~\Anaconda3\envs\MLPR\lib\site-packages\sklearn\ensemble\\_forest.py in ?(self, X, y, sample\_weight)  344 """  345 # Validate or convert input data  346 if issparse(y):  347 raise ValueError("sparse multilabel-indicator for y is not supported.")  --> 348 X, y = self.\_validate\_data(  349 X, y, multi\_output=True, accept\_sparse="csc", dtype=DTYPE  350 )  351 if sample\_weight is not None:  ~\Anaconda3\envs\MLPR\lib\site-packages\sklearn\base.py in ?(self, X, y, reset, validate\_separately, cast\_to\_ndarray, \*\*check\_params)  618 if "estimator" not in check\_y\_params:  619 check\_y\_params = {\*\*default\_check\_params, \*\*check\_y\_params}  620 y = check\_array(y, input\_name="y", \*\*check\_y\_params)  621 else:  --> 622 X, y = check\_X\_y(X, y, \*\*check\_params)  623 out = X, y  624  625 if not no\_val\_X and check\_params.get("ensure\_2d", True):  ~\Anaconda3\envs\MLPR\lib\site-packages\sklearn\utils\validation.py in ?(X, y, accept\_sparse, accept\_large\_sparse, dtype, order, copy, force\_all\_finite, ensure\_2d, allow\_nd, multi\_output, ensure\_min\_samples, ensure\_min\_features, y\_numeric, estimator)  1142 raise ValueError(  1143 f"{estimator\_name} requires y to be passed, but the target y is None"  1144 )  1145  -> 1146 X = check\_array(  1147 X,  1148 accept\_sparse=accept\_sparse,  1149 accept\_large\_sparse=accept\_large\_sparse,  ~\Anaconda3\envs\MLPR\lib\site-packages\sklearn\utils\validation.py in ?(array, accept\_sparse, accept\_large\_sparse, dtype, order, copy, force\_all\_finite, ensure\_2d, allow\_nd, ensure\_min\_samples, ensure\_min\_features, estimator, input\_name)  913 array = xp.astype(array, dtype, copy=False)  914 else:  915 array = \_asarray\_with\_order(array, order=order, dtype=dtype, xp=xp)  916 except ComplexWarning as complex\_warning:  --> 917 raise ValueError(  918 "Complex data not supported\n{}\n".format(array)  919 ) from complex\_warning  920  ~\Anaconda3\envs\MLPR\lib\site-packages\sklearn\utils\\_array\_api.py in ?(array, dtype, order, copy, xp)  376 # Use NumPy API to support order  377 if copy is True:  378 array = numpy.array(array, order=order, dtype=dtype)  379 else:  --> 380 array = numpy.asarray(array, order=order, dtype=dtype)  381  382 # At this point array is a NumPy ndarray. We convert it to an array  383 # container that is consistent with the input's namespace.  ~\Anaconda3\envs\MLPR\lib\site-packages\pandas\core\generic.py in ?(self, dtype)  1996 def \_\_array\_\_(self, dtype: npt.DTypeLike | None = None) -> np.ndarray:  1997 values = self.\_values  -> 1998 arr = np.asarray(values, dtype=dtype)  1999 if (  2000 astype\_is\_view(values.dtype, arr.dtype)  2001 and using\_copy\_on\_write()  ValueError: could not convert string to float: 'W' |
| Final Step | import pandas as pd  from sklearn.ensemble import RandomForestClassifier  from sklearn.metrics import roc\_auc\_score  from sklearn.model\_selection import train\_test\_split  # Load train\_data and test\_data (assuming they are already loaded)  # Define features and target variable  X = train\_data.drop(columns=['TransactionID', 'isFraud'])  y = train\_data['isFraud']  X\_test = test\_data.drop(columns=['TransactionID'])  # Ensure consistency in column names  X.columns = X.columns.str.replace('-', '\_', regex=False)  X\_test.columns = X\_test.columns.str.replace('-', '\_', regex=False)  # Identify non-numeric columns  non\_numeric\_columns\_X = X.select\_dtypes(include=['object']).columns  non\_numeric\_columns\_X\_test = X\_test.select\_dtypes(include=['object']).columns  print(f"Non-numeric columns in X: {non\_numeric\_columns\_X}")  print(f"Non-numeric columns in X\_test: {non\_numeric\_columns\_X\_test}")  # Convert categorical features to numeric  for feature in non\_numeric\_columns\_X:  X[feature] = X[feature].astype('category').cat.codes  X\_test[feature] = X\_test[feature].astype('category').cat.codes  # Fill remaining missing values with -999  X.fillna(-999, inplace=True)  X\_test.fillna(-999, inplace=True)  # Check if columns in X\_test match those in X  if not X.columns.equals(X\_test.columns):  missing\_cols = [col for col in X.columns if col not in X\_test.columns]  extra\_cols = [col for col in X\_test.columns if col not in X.columns]    if missing\_cols:  print(f"Missing columns in X\_test: {missing\_cols}")  if extra\_cols:  print(f"Extra columns in X\_test: {extra\_cols}")  else:  print("Columns in X and X\_test match.")  # Split the training data into training and validation sets  X\_train, X\_val, y\_train, y\_val = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  # Initialize and train the model  model = RandomForestClassifier(n\_estimators=100, random\_state=42)  model.fit(X\_train, y\_train)  # Validate the model  y\_val\_pred = model.predict\_proba(X\_val)[:, 1]  auc\_score = roc\_auc\_score(y\_val, y\_val\_pred)  print(f'Validation AUC Score: {auc\_score}')  # Predict on the test set  test\_predictions = model.predict\_proba(X\_test)[:, 1]  # Prepare submission DataFrame  submission = pd.DataFrame({  'TransactionID': test\_data['TransactionID'],  'isFraud': test\_predictions  })  # Save to CSV  submission.to\_csv('submission.csv', index=False) |