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	0	-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)