```
import pandas as pd
import numpy as np
df=pd.read_excel(r"D:\download\data (1).xlsx")
x=df.iloc[:,0:6].values
y=df.iloc[:,18].values

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,rando
m_state=0)

y_train_log = np.log1p(y_train)
y_test_log = np.log1p(y_test)

from sklearn.preprocessing import OrdinalEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
transformer = ColumnTransformer(transformers=[
    ('encoder1', OrdinalEncoder(categories=[['Tier_3', 'Tier_2',
'Tier_1']]), [4]),
    ('encoder2', OneHotEncoder(drop="first"), [3])
], remainder='passthrough')

# Transform training and test data
X_train_transformed = transformer.fit_transform(x_train)
X_test_transformed = transformer.transform(x_test)

from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score, mean_squared_error
import pandas as pd
import matplotlib.pyplot as plt

# 1. Train Random Forest model
rf = RandomForestRegressor(n_estimators=100, random_state=0)
rf.fit(X_train_transformed, y_train)

# 2. Predict
y_pred_rf = rf.predict(X_test_transformed)

# 3. Evaluate
r2_rf = r2_score(y_test, y_pred_rf)
rmse_rf = mean_squared_error(y_test, y_pred_rf, squared=False)

print("□ Random Forest R² Score:", r2_rf)
print("□ Random Forest RMSE:", rmse_rf)
```

```
□ Random Forest R² Score: 0.9600069460704982
□ Random Forest RMSE: 1472.2988738032293
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics\
_regression.py:492: FutureWarning: 'squared' is deprecated in version
1.4 and will be removed in 1.6. To calculate the root mean squared
```

```
error, use the function'root_mean_squared_error'.
  warnings.warn(



C:\Users\ADMIN\AppData\Local\Temp\ipykernel_15248\2788321332.py:7:
RuntimeWarning: overflow encountered in expm1
  y_pred_original = np.expm1(y_pred_log)

---------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
Cell In[12], line 10
      8 y_pred = model.predict(X_test_scaled)
      9 from sklearn.metrics import mean_squared_error,
mean_absolute_error, r2_score
---> 10 mse = mean_squared_error(y_test, y_pred_original)
     11 mae = mean_absolute_error(y_test, y_pred_original)
     12 r2 = r2_score(y_test, y_pred_original)

File C:\ProgramData\anaconda3\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:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics\
_regression.py:506, in mean_squared_error(y_true, y_pred,
sample_weight, multioutput, squared)
    501     if not squared:
```

```
    502         return root_mean_squared_error(
    503             y_true, y_pred, sample_weight=sample_weight,
multioutput=multioutput
    504         )
--> 506 y_type, y_true, y_pred, multioutput = _check_reg_targets(
    507     y_true, y_pred, multioutput
    508 )
    509 check_consistent_length(y_true, y_pred, sample_weight)
    510 output_errors = np.average((y_true - y_pred) ** 2, axis=0,
weights=sample_weight)

File C:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics\
_regression.py:113, in _check_reg_targets(y_true, y_pred, multioutput,
dtype, xp)
    111 check_consistent_length(y_true, y_pred)
    112 y_true = check_array(y_true, ensure_2d=False, dtype=dtype)
--> 113 y_pred = check_array(y_pred, ensure_2d=False, dtype=dtype)
    115 if y_true.ndim == 1:
    116     y_true = xp.reshape(y_true, (-1, 1))

File C:\ProgramData\anaconda3\Lib\site-packages\sklearn\utils\
validation.py:1064, in check_array(array, accept_sparse,
accept_large_sparse, dtype, order, copy, force_writeable,
force_all_finite, ensure_2d, allow_nd, ensure_min_samples,
ensure_min_features, estimator, input_name)
   1058     raise ValueError(
   1059         "Found array with dim %d. %s expected <= 2."
   1060         % (array.ndim, estimator_name)
   1061     )
   1063 if force_all_finite:
-> 1064     _assert_all_finite(
   1065         array,
   1066         input_name=input_name,
   1067         estimator_name=estimator_name,
   1068         allow_nan=force_all_finite == "allow-nan",
   1069     )
   1071 if copy:
   1072     if _is_numpy_namespace(xp):
   1073         # only make a copy if `array` and `array_orig` may
share memory`

File C:\ProgramData\anaconda3\Lib\site-packages\sklearn\utils\
validation.py:123, in _assert_all_finite(X, allow_nan, msg_dtype,
estimator_name, input_name)
    120 if first_pass_isfinite:
    121     return
--> 123 _assert_all_finite_element_wise(
    124     X,
    125     xp=xp,
    126     allow_nan=allow_nan,
```

```
    127     msg_dtype=msg_dtype,
    128     estimator_name=estimator_name,
    129     input_name=input_name,
    130 )

File C:\ProgramData\anaconda3\Lib\site-packages\sklearn\utils\
validation.py:172, in _assert_all_finite_element_wise(X, xp,
allow_nan, msg_dtype, estimator_name, input_name)
    155 if estimator_name and input_name == "X" and has_nan_error:
    156     # Improve the error message on how to handle missing
values in
    157     # scikit-learn.
    158     msg_err += (
    159         f"\n{estimator_name} does not accept missing values"
    160         " encoded as NaN natively. For supervised learning,
you might want"
   (...)
    170         "#estimators-that-handle-nan-values"
    171     )
--> 172 raise ValueError(msg_err)

ValueError: Input contains infinity or a value too large for
dtype('float64').
```