data-processing-ds-project1

January 2, 2024

Data Pre-processing

Type casting

[117]:
```
import pandas as pd
```

[118]:
```
project = pd.read_csv(r"/content/Datasets.csv")
```

[119]:
```
project.dtypes
```

[119]:
```
Year                      int64
Month                     int64
DayofMonth                int64
DayOfWeek                 int64
Actual_Shipment_Time    float64
Planned_Shipment_Time     int64
Planned_Delivery_Time     int64
Carrier_Name             object
Carrier_Num               int64
Planned_TimeofTravel      int64
Shipment_Delay          float64
Source                   object
Destination              object
Distance                  int64
Delivery_Status         float64
dtype: object
```

[ ]:
```
help(project.astype)
```

```
Help on method astype in module pandas.core.generic:

astype(dtype, copy: 'bool_t' = True, errors: 'IgnoreRaise' = 'raise') ->
'NDFrameT' method of pandas.core.frame.DataFrame instance
    Cast a pandas object to a specified dtype ``dtype``.

    Parameters
    ----------
    dtype : data type, or dict of column name -> data type
        Use a numpy.dtype or Python type to cast entire pandas object to
```

```
    the same type. Alternatively, use {col: dtype, …}, where col is a
    column label and dtype is a numpy.dtype or Python type to cast one
    or more of the DataFrame's columns to column-specific types.
copy : bool, default True
    Return a copy when ``copy=True`` (be very careful setting
    ``copy=False`` as changes to values then may propagate to other
    pandas objects).
errors : {'raise', 'ignore'}, default 'raise'
    Control raising of exceptions on invalid data for provided dtype.

    - ``raise`` : allow exceptions to be raised
    - ``ignore`` : suppress exceptions. On error return original object.

Returns
-------
casted : same type as caller

See Also
--------
to_datetime : Convert argument to datetime.
to_timedelta : Convert argument to timedelta.
to_numeric : Convert argument to a numeric type.
numpy.ndarray.astype : Cast a numpy array to a specified type.

Notes
-----
.. deprecated:: 1.3.0

    Using ``astype`` to convert from timezone-naive dtype to
    timezone-aware dtype is deprecated and will raise in a
    future version.  Use :meth:`Series.dt.tz_localize` instead.

Examples
--------
Create a DataFrame:

>>> d = {'col1': [1, 2], 'col2': [3, 4]}
>>> df = pd.DataFrame(data=d)
>>> df.dtypes
col1    int64
col2    int64
dtype: object

Cast all columns to int32:

>>> df.astype('int32').dtypes
col1    int32
col2    int32
```

```
dtype: object

Cast col1 to int32 using a dictionary:

>>> df.astype({'col1': 'int32'}).dtypes
col1    int32
col2    int64
dtype: object

Create a series:

>>> ser = pd.Series([1, 2], dtype='int32')
>>> ser
0    1
1    2
dtype: int32
>>> ser.astype('int64')
0    1
1    2
dtype: int64

Convert to categorical type:

>>> ser.astype('category')
0    1
1    2
dtype: category
Categories (2, int64): [1, 2]

Convert to ordered categorical type with custom ordering:

>>> from pandas.api.types import CategoricalDtype
>>> cat_dtype = CategoricalDtype(
…     categories=[2, 1], ordered=True)
>>> ser.astype(cat_dtype)
0    1
1    2
dtype: category
Categories (2, int64): [2 < 1]

Note that using ``copy=False`` and changing data on a new
pandas object may propagate changes:

>>> s1 = pd.Series([1, 2])
>>> s2 = s1.astype('int64', copy=False)
>>> s2[0] = 10
>>> s1  # note that s1[0] has changed too
0    10
```

```
1     2
dtype: int64

Create a series of dates:

>>> ser_date = pd.Series(pd.date_range('20200101', periods=3))
>>> ser_date
0   2020-01-01
1   2020-01-02
2   2020-01-03
dtype: datetime64[ns]
```

1 Convert 'int64' to 'str' (string) type.

[ ]:
```
project.Year = project.Year.astype('str')
```

[ ]:
```
project.dtypes
```

[ ]:
```
Year                      object
Month                      int64
DayofMonth                 int64
DayOfWeek                  int64
Actual_Shipment_Time     float64
Planned_Shipment_Time      int64
Planned_Delivery_Time      int64
Carrier_Name              object
Carrier_Num                int64
Planned_TimeofTravel       int64
Shipment_Delay           float64
Source                    object
Destination               object
Distance                   int64
Delivery_Status          float64
dtype: object
```

[ ]:
```
project.Month = project.Month.astype('str')
```

[ ]:
```
project.dtypes
```

[ ]:
```
Year                      object
Month                     object
DayofMonth                 int64
DayOfWeek                  int64
Actual_Shipment_Time     float64
Planned_Shipment_Time      int64
```

```
Planned_Delivery_Time      int64
Carrier_Name              object
Carrier_Num                int64
Planned_TimeofTravel       int64
Shipment_Delay           float64
Source                    object
Destination               object
Distance                   int64
Delivery_Status          float64
dtype: object
```

[ ]:
```
project.Planned_Shipment_Time = project.Planned_Shipment_Time.astype('str')
```

[ ]:
```
project.dtypes
```

[ ]:
```
Year                      object
Month                     object
DayofMonth                 int64
DayOfWeek                  int64
Actual_Shipment_Time     float64
Planned_Shipment_Time     object
Planned_Delivery_Time      int64
Carrier_Name              object
Carrier_Num                int64
Planned_TimeofTravel       int64
Shipment_Delay           float64
Source                    object
Destination               object
Distance                   int64
Delivery_Status          float64
dtype: object
```

2 convert 'str' to 'int64' type.

[ ]:
```
project.Year = project.Year.astype('int64')
```

[ ]:
```
project.dtypes
```

[ ]:
```
Year                       int64
Month                     object
DayofMonth                 int64
DayOfWeek                  int64
Actual_Shipment_Time     float64
Planned_Shipment_Time     object
Planned_Delivery_Time      int64
Carrier_Name              object
```

```
Carrier_Num                int64
Planned_TimeofTravel       int64
Shipment_Delay           float64
Source                    object
Destination               object
Distance                   int64
Delivery_Status          float64
dtype: object
```

[ ]:
```
project.Month = project.Month.astype('int64')
```

[ ]:
```
project.dtypes
```

[ ]:
```
Year                       int64
Month                      int64
DayofMonth                 int64
DayOfWeek                  int64
Actual_Shipment_Time     float64
Planned_Shipment_Time     object
Planned_Delivery_Time      int64
Carrier_Name              object
Carrier_Num                int64
Planned_TimeofTravel       int64
Shipment_Delay           float64
Source                    object
Destination               object
Distance                   int64
Delivery_Status          float64
dtype: object
```

[ ]:
```
project.Planned_Shipment_Time = project.Planned_Shipment_Time.astype('int64')
```

[ ]:
```
project.dtypes
```

[ ]:
```
Year                       int64
Month                      int64
DayofMonth                 int64
DayOfWeek                  int64
Actual_Shipment_Time     float64
Planned_Shipment_Time      int64
Planned_Delivery_Time      int64
Carrier_Name              object
Carrier_Num                int64
Planned_TimeofTravel       int64
Shipment_Delay           float64
Source                    object
Destination               object
```

```
Distance                    int64
Delivery_Status           float64
dtype: object
```

3 'float64' into 'int64' type

[ ]:
```
project.Actual_Shipment_Time = project.Actual_Shipment_Time.astype('int64')
```

```
---------------------------------------------------------------------------
IntCastingNaNError                        Traceback (most recent call last)
<ipython-input-17-a261dbd43cc9> in <cell line: 1>()
----> 1 project.Actual_Shipment_Time = project.Actual_Shipment_Time.
 ↪astype('int64')

/usr/local/lib/python3.10/dist-packages/pandas/core/generic.py in astype(self,␣
 ↪dtype, copy, errors)
   6238         else:
   6239             # else, only a single dtype is given
-> 6240             new_data = self._mgr.astype(dtype=dtype, copy=copy,␣
 ↪errors=errors)
   6241             return self._constructor(new_data).__finalize__(self,␣
 ↪method="astype")
   6242

/usr/local/lib/python3.10/dist-packages/pandas/core/internals/managers.py in␣
 ↪astype(self, dtype, copy, errors)
    446
    447     def astype(self: T, dtype, copy: bool = False, errors: str =␣
 ↪"raise") -> T:
--> 448         return self.apply("astype", dtype=dtype, copy=copy,␣
 ↪errors=errors)
    449
    450     def convert(

/usr/local/lib/python3.10/dist-packages/pandas/core/internals/managers.py in␣
 ↪apply(self, f, align_keys, ignore_failures, **kwargs)
    350                     applied = b.apply(f, **kwargs)
    351                 else:
--> 352                     applied = getattr(b, f)(**kwargs)
    353             except (TypeError, NotImplementedError):
    354                 if not ignore_failures:

/usr/local/lib/python3.10/dist-packages/pandas/core/internals/blocks.py in␣
 ↪astype(self, dtype, copy, errors)
    524         values = self.values
    525
```

```
--> 526         new_values = astype_array_safe(values, dtype, copy=copy,␣
 ↪errors=errors)
    527
    528         new_values = maybe_coerce_values(new_values)

/usr/local/lib/python3.10/dist-packages/pandas/core/dtypes/astype.py in␣
 ↪astype_array_safe(values, dtype, copy, errors)
    297
    298     try:
--> 299         new_values = astype_array(values, dtype, copy=copy)
    300     except (ValueError, TypeError):
    301         # e.g. astype_nansafe can fail on object-dtype of strings

/usr/local/lib/python3.10/dist-packages/pandas/core/dtypes/astype.py in␣
 ↪astype_array(values, dtype, copy)
    228
    229     else:
--> 230         values = astype_nansafe(values, dtype, copy=copy)
    231
    232     # in pandas we don't store numpy str dtypes, so convert to object

/usr/local/lib/python3.10/dist-packages/pandas/core/dtypes/astype.py in␣
 ↪astype_nansafe(arr, dtype, copy, skipna)
    138
    139     elif np.issubdtype(arr.dtype, np.floating) and␣
 ↪is_integer_dtype(dtype):
--> 140         return _astype_float_to_int_nansafe(arr, dtype, copy)
    141
    142     elif is_object_dtype(arr.dtype):

/usr/local/lib/python3.10/dist-packages/pandas/core/dtypes/astype.py in␣
 ↪_astype_float_to_int_nansafe(values, dtype, copy)
    180     """
    181     if not np.isfinite(values).all():
--> 182         raise IntCastingNaNError(
    183             "Cannot convert non-finite values (NA or inf) to integer"
    184         )

IntCastingNaNError: Cannot convert non-finite values (NA or inf) to integer
```

[ ]:
```
project.dtypes
```

[ ]:
```
Year                        int64
Month                       int64
DayofMonth                  int64
DayOfWeek                   int64
```

```
Actual_Shipment_Time     float64
Planned_Shipment_Time      int64
Planned_Delivery_Time      int64
Carrier_Name              object
Carrier_Num                int64
Planned_TimeofTravel       int64
Shipment_Delay           float64
Source                    object
Destination               object
Distance                   int64
Delivery_Status          float64
dtype: object
```

[ ]:
```
project.Shipment_Delay = project.Shipment_Delay.astype('int64')
```

```
---------------------------------------------------------------------------
IntCastingNaNError                        Traceback (most recent call last)
<ipython-input-19-ab5ba5d19076> in <cell line: 1>()
----> 1 project.Shipment_Delay = project.Shipment_Delay.astype('int64')

/usr/local/lib/python3.10/dist-packages/pandas/core/generic.py in astype(self,␣
↪dtype, copy, errors)
   6238         else:
   6239             # else, only a single dtype is given
-> 6240             new_data = self._mgr.astype(dtype=dtype, copy=copy,␣
↪errors=errors)
   6241             return self._constructor(new_data).__finalize__(self,␣
↪method="astype")
   6242

/usr/local/lib/python3.10/dist-packages/pandas/core/internals/managers.py in␣
↪astype(self, dtype, copy, errors)
    446
    447     def astype(self: T, dtype, copy: bool = False, errors: str =␣
↪"raise") -> T:
--> 448         return self.apply("astype", dtype=dtype, copy=copy,␣
↪errors=errors)
    449
    450     def convert(

/usr/local/lib/python3.10/dist-packages/pandas/core/internals/managers.py in␣
↪apply(self, f, align_keys, ignore_failures, **kwargs)
    350                     applied = b.apply(f, **kwargs)
    351                 else:
--> 352                     applied = getattr(b, f)(**kwargs)
    353             except (TypeError, NotImplementedError):
    354                 if not ignore_failures:
```

```
/usr/local/lib/python3.10/dist-packages/pandas/core/internals/blocks.py in␣
↪astype(self, dtype, copy, errors)
    524         values = self.values
    525
--> 526         new_values = astype_array_safe(values, dtype, copy=copy,␣
↪errors=errors)
    527
    528         new_values = maybe_coerce_values(new_values)

/usr/local/lib/python3.10/dist-packages/pandas/core/dtypes/astype.py in␣
↪astype_array_safe(values, dtype, copy, errors)
    297
    298     try:
--> 299         new_values = astype_array(values, dtype, copy=copy)
    300     except (ValueError, TypeError):
    301         # e.g. astype_nansafe can fail on object-dtype of strings

/usr/local/lib/python3.10/dist-packages/pandas/core/dtypes/astype.py in␣
↪astype_array(values, dtype, copy)
    228
    229     else:
--> 230         values = astype_nansafe(values, dtype, copy=copy)
    231
    232     # in pandas we don't store numpy str dtypes, so convert to object

/usr/local/lib/python3.10/dist-packages/pandas/core/dtypes/astype.py in␣
↪astype_nansafe(arr, dtype, copy, skipna)
    138
    139     elif np.issubdtype(arr.dtype, np.floating) and␣
↪is_integer_dtype(dtype):
--> 140         return _astype_float_to_int_nansafe(arr, dtype, copy)
    141
    142     elif is_object_dtype(arr.dtype):

/usr/local/lib/python3.10/dist-packages/pandas/core/dtypes/astype.py in␣
↪_astype_float_to_int_nansafe(values, dtype, copy)
    180     """
    181     if not np.isfinite(values).all():
--> 182         raise IntCastingNaNError(
    183             "Cannot convert non-finite values (NA or inf) to integer"
    184         )

IntCastingNaNError: Cannot convert non-finite values (NA or inf) to integer
```

[ ]:
```
project.dtypes
```

[ ]:
```
Year                        int64
Month                       int64
DayofMonth                  int64
DayOfWeek                   int64
Actual_Shipment_Time      float64
Planned_Shipment_Time       int64
Planned_Delivery_Time       int64
Carrier_Name               object
Carrier_Num                 int64
Planned_TimeofTravel        int64
Shipment_Delay            float64
Source                     object
Destination                object
Distance                    int64
Delivery_Status           float64
dtype: object
```

3.0.1 Identify duplicate records in the data

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

[ ]:
```
project = pd.read_csv(r"/content/Datasets.csv")
```

4 Duplicates in rows

[ ]:
```
help(project.duplicated)
```

```
Help on method duplicated in module pandas.core.frame:

duplicated(subset: 'Hashable | Sequence[Hashable] | None' = None, keep:
"Literal['first', 'last', False]" = 'first') -> 'Series' method of
pandas.core.frame.DataFrame instance
    Return boolean Series denoting duplicate rows.

    Considering certain columns is optional.

    Parameters
    ----------
    subset : column label or sequence of labels, optional
        Only consider certain columns for identifying duplicates, by
        default use all of the columns.
    keep : {'first', 'last', False}, default 'first'
        Determines which duplicates (if any) to mark.

        - ``first`` : Mark duplicates as ``True`` except for the first
occurrence.
```

```
    - ``last`` : Mark duplicates as ``True`` except for the last occurrence.
    - False : Mark all duplicates as ``True``.

Returns
-------
Series
    Boolean series for each duplicated rows.

See Also
--------
Index.duplicated : Equivalent method on index.
Series.duplicated : Equivalent method on Series.
Series.drop_duplicates : Remove duplicate values from Series.
DataFrame.drop_duplicates : Remove duplicate values from DataFrame.

Examples
--------
Consider dataset containing ramen rating.

>>> df = pd.DataFrame({
…     'brand': ['Yum Yum', 'Yum Yum', 'Indomie', 'Indomie', 'Indomie'],
…     'style': ['cup', 'cup', 'cup', 'pack', 'pack'],
…     'rating': [4, 4, 3.5, 15, 5]
… })
>>> df
    brand style  rating
0  Yum Yum   cup     4.0
1  Yum Yum   cup     4.0
2  Indomie   cup     3.5
3  Indomie  pack    15.0
4  Indomie  pack     5.0

By default, for each set of duplicated values, the first occurrence
is set on False and all others on True.

>>> df.duplicated()
0    False
1     True
2    False
3    False
4    False
dtype: bool

By using 'last', the last occurrence of each set of duplicated values
is set on False and all others on True.

>>> df.duplicated(keep='last')
0     True
```

```
1    False
2    False
3    False
4    False
dtype: bool

By setting ``keep`` on False, all duplicates are True.

>>> df.duplicated(keep=False)
0     True
1     True
2    False
3    False
4    False
dtype: bool

To find duplicates on specific column(s), use ``subset``.

>>> df.duplicated(subset=['brand'])
0    False
1     True
2    False
3     True
4     True
dtype: bool
```

[ ]:
```
duplicate = project.duplicated()
```

[ ]:
```
duplicate
```

[ ]:
```
0       False
1       False
2       False
3       False
4       False
        …
7994    False
7995    False
7996    False
7997    False
7998    False
Length: 7999, dtype: bool
```

[ ]:
```
sum(duplicate)
```

[ ]:
```
0
```

5 Duplicates in rows

[ ]:
```
help(project.duplicated)
```

```
Help on method duplicated in module pandas.core.frame:

duplicated(subset: 'Hashable | Sequence[Hashable] | None' = None, keep:
"Literal['first', 'last', False]" = 'first') -> 'Series' method of
pandas.core.frame.DataFrame instance
    Return boolean Series denoting duplicate rows.

    Considering certain columns is optional.

    Parameters
    ----------
    subset : column label or sequence of labels, optional
        Only consider certain columns for identifying duplicates, by
        default use all of the columns.
    keep : {'first', 'last', False}, default 'first'
        Determines which duplicates (if any) to mark.

        - ``first`` : Mark duplicates as ``True`` except for the first
occurrence.
        - ``last`` : Mark duplicates as ``True`` except for the last occurrence.
        - False : Mark all duplicates as ``True``.

    Returns
    -------
    Series
        Boolean series for each duplicated rows.

    See Also
    --------
    Index.duplicated : Equivalent method on index.
    Series.duplicated : Equivalent method on Series.
    Series.drop_duplicates : Remove duplicate values from Series.
    DataFrame.drop_duplicates : Remove duplicate values from DataFrame.

    Examples
    --------
    Consider dataset containing ramen rating.

    >>> df = pd.DataFrame({
    …     'brand': ['Yum Yum', 'Yum Yum', 'Indomie', 'Indomie', 'Indomie'],
    …     'style': ['cup', 'cup', 'cup', 'pack', 'pack'],
    …     'rating': [4, 4, 3.5, 15, 5]
    … })
```

```
>>> df
    brand style  rating
0  Yum Yum   cup     4.0
1  Yum Yum   cup     4.0
2  Indomie   cup     3.5
3  Indomie  pack    15.0
4  Indomie  pack     5.0

By default, for each set of duplicated values, the first occurrence
is set on False and all others on True.

>>> df.duplicated()
0    False
1     True
2    False
3    False
4    False
dtype: bool

By using 'last', the last occurrence of each set of duplicated values
is set on False and all others on True.

>>> df.duplicated(keep='last')
0     True
1    False
2    False
3    False
4    False
dtype: bool

By setting ``keep`` on False, all duplicates are True.

>>> df.duplicated(keep=False)
0     True
1     True
2    False
3    False
4    False
dtype: bool

To find duplicates on specific column(s), use ``subset``.

>>> df.duplicated(subset=['brand'])
0    False
1     True
2    False
3     True
4     True
```

```
dtype: bool
```

[ ]:
```
duplicate = project.duplicated()  # Returns Boolean Series denoting duplicate␣
↪rows.
```

[ ]:
```
duplicate
```

[ ]:
```
0       False
1       False
2       False
3       False
4       False
        …
7994    False
7995    False
7996    False
7997    False
7998    False
Length: 7999, dtype: bool
```

[ ]:
```
sum(duplicate)
```

[ ]: 0

6 Parameters

[ ]:
```
duplicate = project.duplicated(keep = 'last')
```

[ ]:
```
duplicate
```

[ ]:
```
0       False
1       False
2       False
3       False
4       False
        …
7994    False
7995    False
7996    False
7997    False
7998    False
Length: 7999, dtype: bool
```

[ ]:
```
duplicate = project.duplicated(keep = False)
```

[ ]:
```
duplicate
```

[ ]:
```
0       False
1       False
2       False
3       False
4       False
        …
7994    False
7995    False
7996    False
7997    False
7998    False
Length: 7999, dtype: bool
```

7 Removing Duplicates

[ ]:
```
project1 = project.drop_duplicates() # Returns DataFrame with duplicate rows␣
↪removed.
```

8 Parameters

[ ]:
```
project1 = project.drop_duplicates(keep = 'last')
```

[ ]:
```
project1 = project.drop_duplicates(keep = False)
```

9 Duplicates in Columns

10 We can use correlation coefficient values to identify columns which have duplicate information

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

[ ]:
```
project = pd.read_csv(r"/content/Datasets.csv")
```

11 Correlation coefficient

[ ]:
```
project.corr()
```

```
<ipython-input-40-f63a631e511a>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
```

```
to silence this warning.
  project.corr()
```

[ ]:

	Year	Month	DayofMonth	DayOfWeek	\
Year	NaN	NaN	NaN	NaN	
Month	NaN	NaN	NaN	NaN	
DayofMonth	NaN	NaN	1.000000	1.000000	
DayOfWeek	NaN	NaN	1.000000	1.000000	
Actual_Shipment_Time	NaN	NaN	-0.014877	-0.014877	
Planned_Shipment_Time	NaN	NaN	-0.004550	-0.004550	
Planned_Delivery_Time	NaN	NaN	-0.000644	-0.000644	
Carrier_Num	NaN	NaN	-0.053688	-0.053688	
Planned_TimeofTravel	NaN	NaN	0.022032	0.022032	
Shipment_Delay	NaN	NaN	-0.077483	-0.077483	
Distance	NaN	NaN	0.016220	0.016220	
Delivery_Status	NaN	NaN	-0.121121	-0.121121	

	Actual_Shipment_Time	Planned_Shipment_Time	\
Year	NaN	NaN	
Month	NaN	NaN	
DayofMonth	-0.014877	-0.004550	
DayOfWeek	-0.014877	-0.004550	
Actual_Shipment_Time	1.000000	0.992386	
Planned_Shipment_Time	0.992386	1.000000	
Planned_Delivery_Time	0.847986	0.858210	
Carrier_Num	0.005744	0.005147	
Planned_TimeofTravel	-0.063763	-0.070638	
Shipment_Delay	0.434833	0.338752	
Distance	-0.053634	-0.062261	
Delivery_Status	0.459595	0.397657	

	Planned_Delivery_Time	Carrier_Num	\
Year	NaN	NaN	
Month	NaN	NaN	
DayofMonth	-0.000644	-0.053688	
DayOfWeek	-0.000644	-0.053688	
Actual_Shipment_Time	0.847986	0.005744	
Planned_Shipment_Time	0.858210	0.005147	
Planned_Delivery_Time	1.000000	-0.004370	
Carrier_Num	-0.004370	1.000000	
Planned_TimeofTravel	0.030032	0.045030	
Shipment_Delay	0.270309	0.004711	
Distance	0.038032	0.035700	
Delivery_Status	0.341430	0.005415	

	Planned_TimeofTravel	Shipment_Delay	Distance	\
Year	NaN	NaN	NaN	

```
Month                                 NaN             NaN       NaN
DayofMonth                       0.022032       -0.077483  0.016220
DayOfWeek                        0.022032       -0.077483  0.016220
Actual_Shipment_Time            -0.063763        0.434833 -0.053634
Planned_Shipment_Time           -0.070638        0.338752 -0.062261
Planned_Delivery_Time            0.030032        0.270309  0.038032
Carrier_Num                      0.045030        0.004711  0.035700
Planned_TimeofTravel             1.000000        0.032342  0.980355
Shipment_Delay                   0.032342        1.000000  0.050998
Distance                         0.980355        0.050998  1.000000
Delivery_Status                  0.025275        0.692433  0.044404

                       Delivery_Status
Year                               NaN
Month                              NaN
DayofMonth                   -0.121121
DayOfWeek                    -0.121121
Actual_Shipment_Time          0.459595
Planned_Shipment_Time         0.397657
Planned_Delivery_Time         0.341430
Carrier_Num                   0.005415
Planned_TimeofTravel          0.025275
Shipment_Delay                0.692433
Distance                      0.044404
Delivery_Status               1.000000
```

Missing Values - Imputation

[ ]:
```python
import numpy as np
```

[ ]:
```python
import pandas as pd
```

[ ]:
```python
project = pd.read_csv(r"/content/Datasets.csv")
```

[ ]:
```python
project.isna().sum()
```

[ ]:
```
Year                      0
Month                     0
DayofMonth                0
DayOfWeek                 0
Actual_Shipment_Time    139
Planned_Shipment_Time     0
Planned_Delivery_Time     0
Carrier_Name              0
Carrier_Num               0
Planned_TimeofTravel      0
Shipment_Delay          139
```

```
Source                     0
Destination                0
Distance                   0
Delivery_Status          139
dtype: int64
```

12 For Mean, Median, Mode imputation we can use Simple Imputer or df.fillna()

[ ]:
```
from sklearn.impute import SimpleImputer
```

13 Mean Imputer

[ ]:
```
mean_imputer1 = SimpleImputer(missing_values = np.nan, strategy = 'mean')
```

[ ]:
```
project["Actual_Shipment_Time"] = pd.DataFrame(mean_imputer1.
 ↪fit_transform(project[["Actual_Shipment_Time"]]))
```

[ ]:
```
project["Actual_Shipment_Time"].isna().sum()
```

[ ]: 0

14 Median Imputer

[ ]:
```
median_imputer1 = SimpleImputer(missing_values = np.nan, strategy = 'median')
```

[ ]:
```
project["Planned_Shipment_Time"] = pd.DataFrame(median_imputer1.
 ↪fit_transform(project[["Planned_Shipment_Time"]]))
```

[ ]:
```
project["Planned_Shipment_Time"].isna().sum()
```

[ ]: 0

[ ]:
```
project.isna().sum()
```

[ ]:
```
Year                       0
Month                      0
DayofMonth                 0
DayOfWeek                  0
Actual_Shipment_Time       0
Planned_Shipment_Time      0
Planned_Delivery_Time      0
Carrier_Name               0
```

```
Carrier_Num                0
Planned_TimeofTravel       0
Shipment_Delay           139
Source                     0
Destination                0
Distance                   0
Delivery_Status          139
dtype: int64
```

15 Mode Imputer

[ ]:
```
mode_imputer1 = SimpleImputer(missing_values = np.nan, strategy =␣
↪'most_frequent')
```

[ ]:
```
project["Planned_Delivery_Time"] = pd.DataFrame(mode_imputer1.
↪fit_transform(project[["Planned_Delivery_Time"]]))
```

[ ]:
```
project["Planned_Delivery_Time"] = pd.DataFrame(mode_imputer1.
↪fit_transform(project[["Planned_Delivery_Time"]]))
```

[ ]:
```
project.isnull().sum()
```

[ ]:
```
Year                       0
Month                      0
DayofMonth                 0
DayOfWeek                  0
Actual_Shipment_Time       0
Planned_Shipment_Time      0
Planned_Delivery_Time      0
Carrier_Name               0
Carrier_Num                0
Planned_TimeofTravel       0
Shipment_Delay           139
Source                     0
Destination                0
Distance                   0
Delivery_Status          139
dtype: int64
```

Outlier Treatment

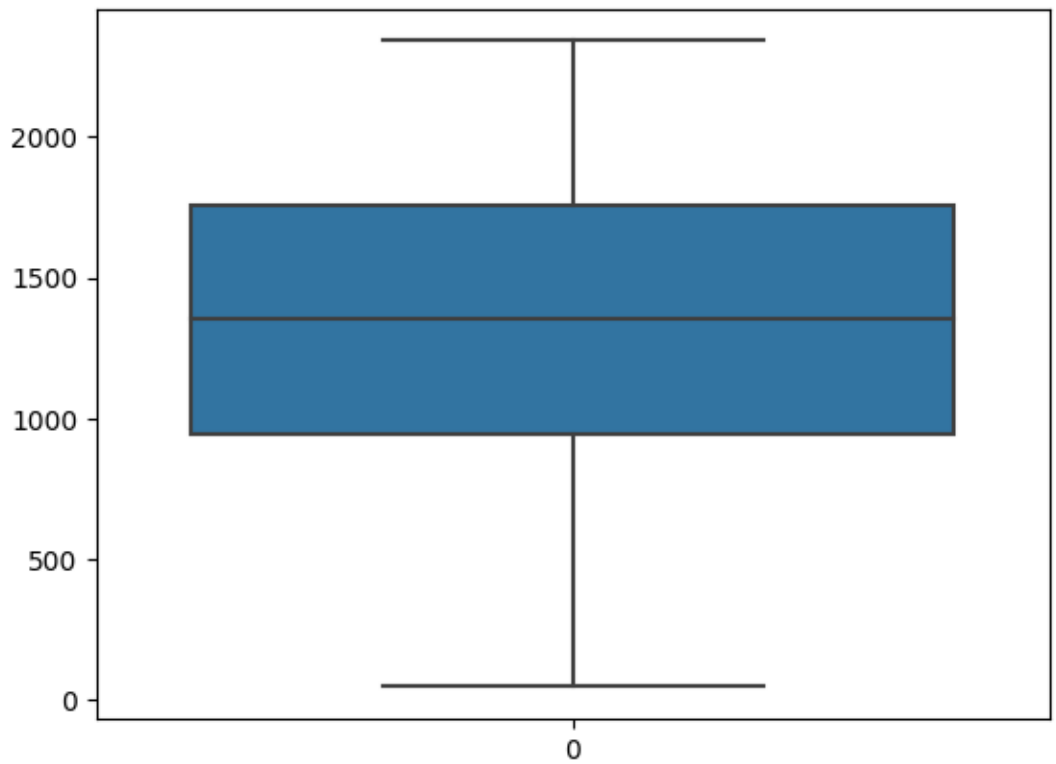
[1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
```

[2]:
```
project = pd.read_csv(r"/content/Datasets.csv")
```

16 Let's find outliers in Actual_Shipment_Time

[3]:
```
sns.boxplot(project.Actual_Shipment_Time)
```

[3]: <Axes: >



17 No outliers in Actual_Shipment_Time column

18 Detection of outliers (find limits for Actual_Shipment_Time based on IQR)

[4]:
```
IQR = project['Actual_Shipment_Time'].quantile(0.75) -␣
↪project['Actual_Shipment_Time'].quantile(0.25)
```

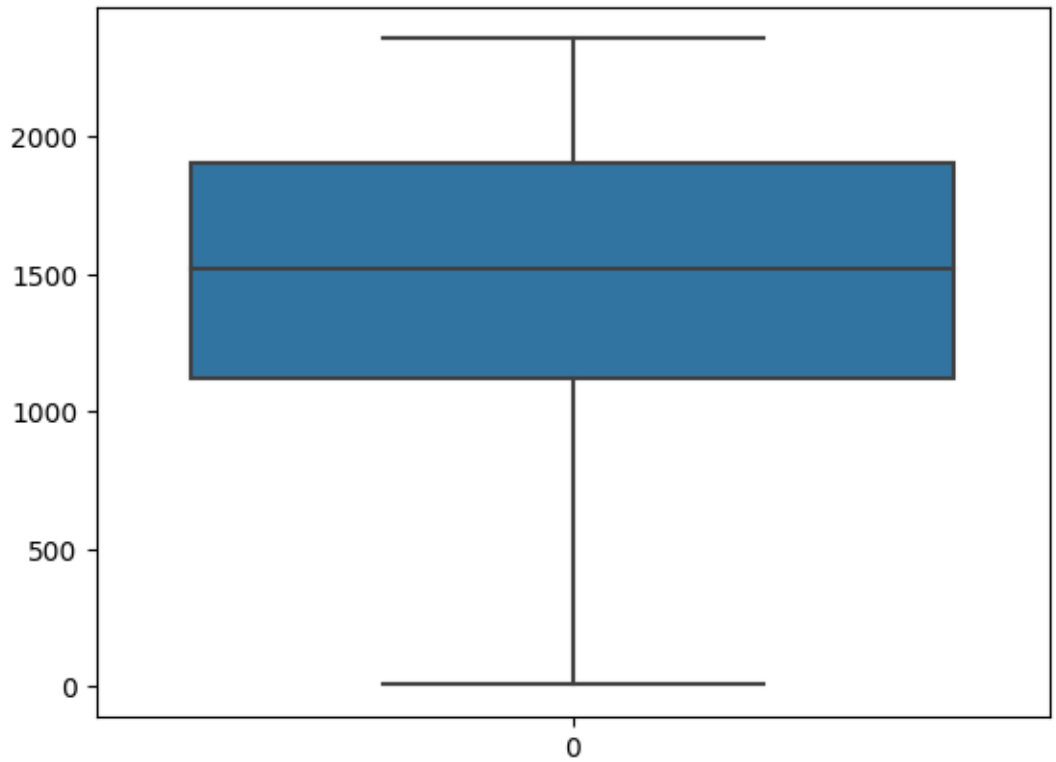
[5]:
```
lower_limit1 = project['Actual_Shipment_Time'].quantile(0.25) - (IQR * 1.5)
```

[6]:
```
upper_limit1 = project['Actual_Shipment_Time'].quantile(0.75) + (IQR * 1.5)
```

19 Let's find outliers in Planned_Delivered_Time

[7]:
```
sns.boxplot(project.Planned_Delivery_Time)
```

[7]: <Axes: >



20 No outliers in Planned_Delivered_Time column

21 Detection of outliers (find limits for Planned_Delivery_Time based on IQR)

[8]:
```
IQR = project['Planned_Delivery_Time'].quantile(0.75) -␣
↪project['Planned_Delivery_Time'].quantile(0.25)
```

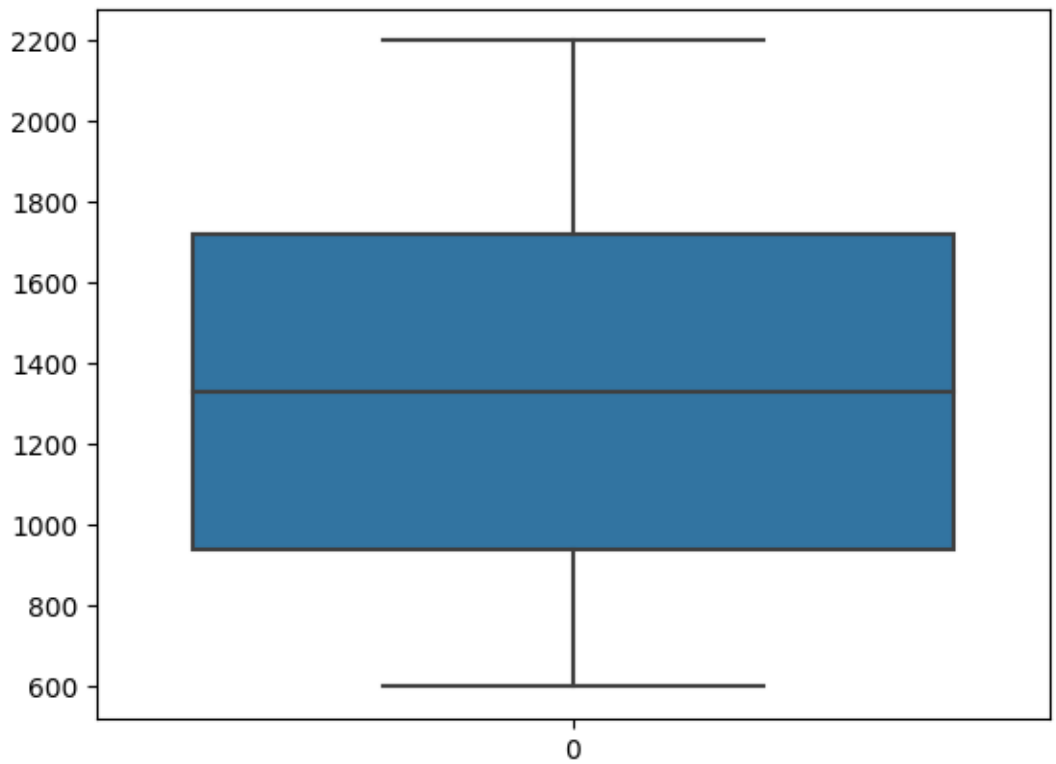
[9]:
```
lower_limit3 = project['Planned_Delivery_Time'].quantile(0.25) - (IQR * 1.5)
```

[10]:
```
upper_limit3 = project['Planned_Delivery_Time'].quantile(0.75) + (IQR * 1.5)
```

22 Let's find outliers in Planned_Shipment_Time

[11]:
```
sns.boxplot(project.Planned_Shipment_Time)
```

[11]: <Axes: >



23 No outliers in Planned_Shipment_Time column

24 Detection of outliers (find limits for Planned_Shipment_Time based on IQR)

[12]:
```
IQR = project['Planned_Shipment_Time'].quantile(0.75) -␣
↪project['Planned_Shipment_Time'].quantile(0.25)
```

[13]:
```
lower_limit2 = project['Planned_Shipment_Time'].quantile(0.25) - (IQR * 1.5)
```

[14]:
```
upper_limit2 = project['Planned_Shipment_Time'].quantile(0.75) + (IQR * 1.5)
```

1. Remove (let's trim the dataset)

25 Trimming Technique

26 Let's flag the outliers in the dataset

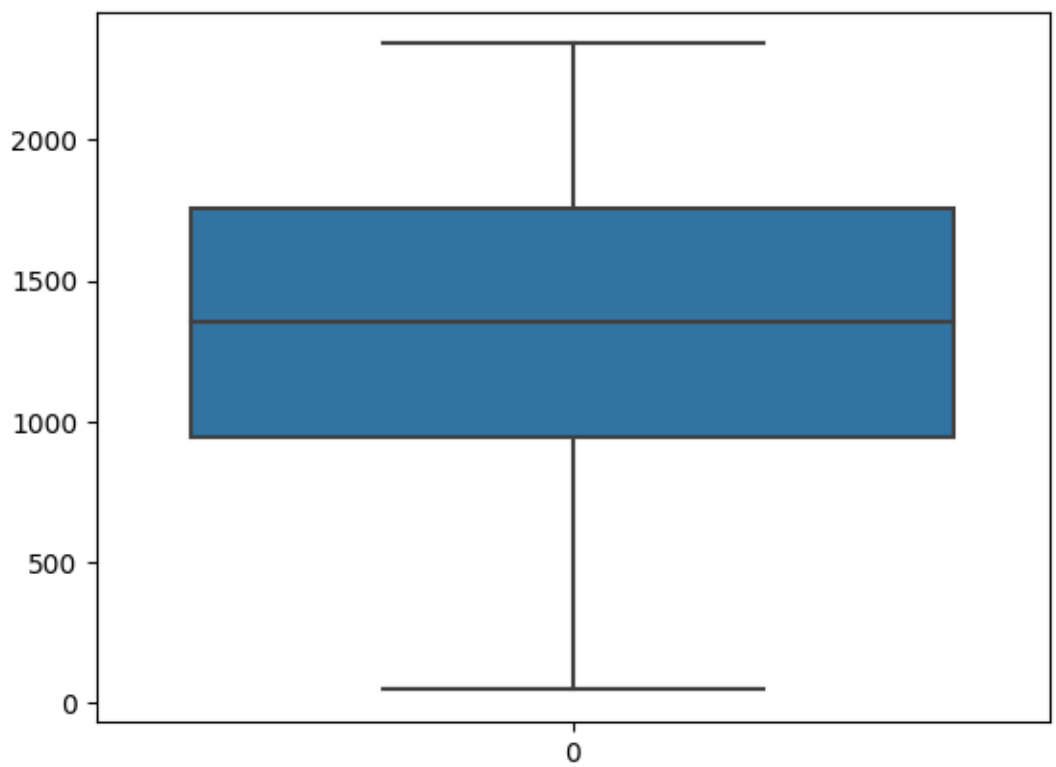
[15]:
```
outliers_project1 = np.where(project.Actual_Shipment_Time > upper_limit1, True,␣
↪np.where(project.Actual_Shipment_Time < lower_limit1, True, False))
```

[16]:
```
# outliers data
project_out1 = project.loc[outliers_project1, ]
project_trimmed1 = project.loc[~(outliers_project1), ]
project.shape, project_trimmed1.shape
```

[16]: ((7999, 15), (7999, 15))

[17]:
```
# Let's explore outliers in the trimmed dataset
sns.boxplot(project_trimmed1.Actual_Shipment_Time)
```

[17]: <Axes: >



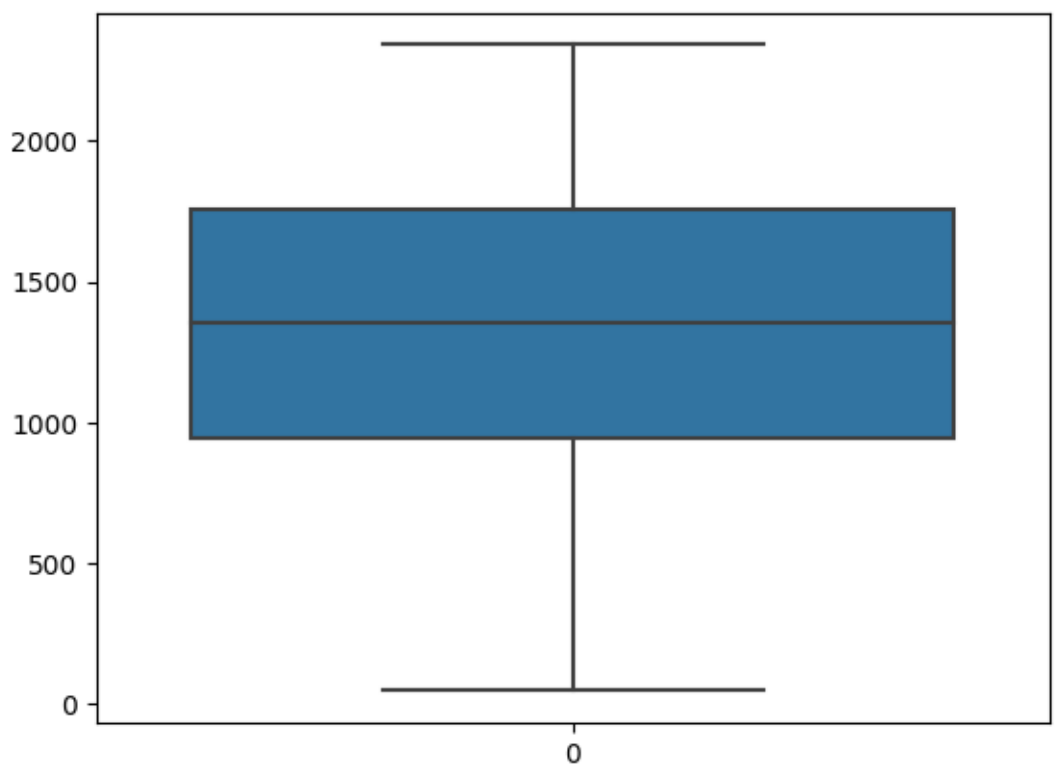
[18]:
```
outliers_project2 = np.where(project.Planned_Shipment_Time > upper_limit1,␣
↪True, np.where(project.Planned_Shipment_Time < lower_limit1, True, False))
```

```
# outliers data
project_out2 = project.loc[outliers_project2, ]
project_trimmed2 = project.loc[~(outliers_project2), ]
project.shape, project_trimmed2.shape
```

[18]: ((7999, 15), (7999, 15))

[19]:
```
# Let's explore outliers in the trimmed dataset
sns.boxplot(project_trimmed2.Actual_Shipment_Time)
```

[19]: <Axes: >

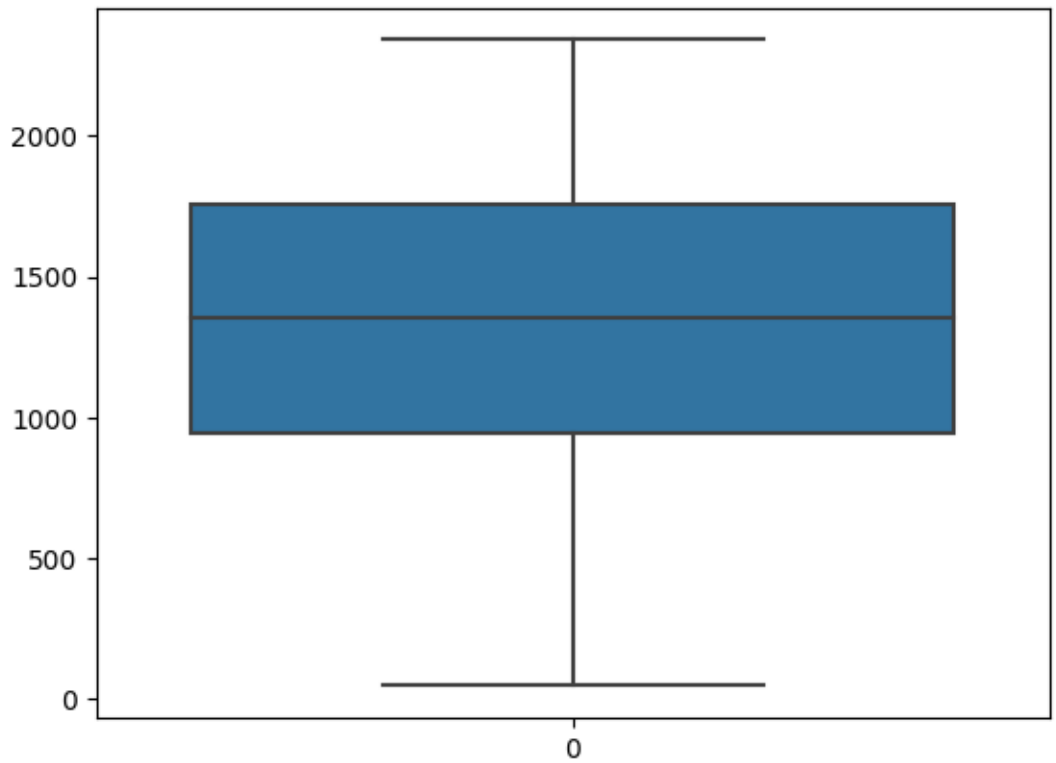


[20]:
```
outliers_project3 = np.where(project.Planned_Delivery_Time > upper_limit1,␣
↪True, np.where(project.Planned_Delivery_Time < lower_limit1, True, False))

# outliers data
project_out3 = project.loc[outliers_project3, ]
project_trimmed3 = project.loc[~(outliers_project3), ]
project.shape, project_trimmed3.shape
```

[20]: ((7999, 15), (7999, 15))

[21]:
```
# Let's explore outliers in the trimmed dataset
sns.boxplot(project_trimmed3.Actual_Shipment_Time)
```

[21]: <Axes: >

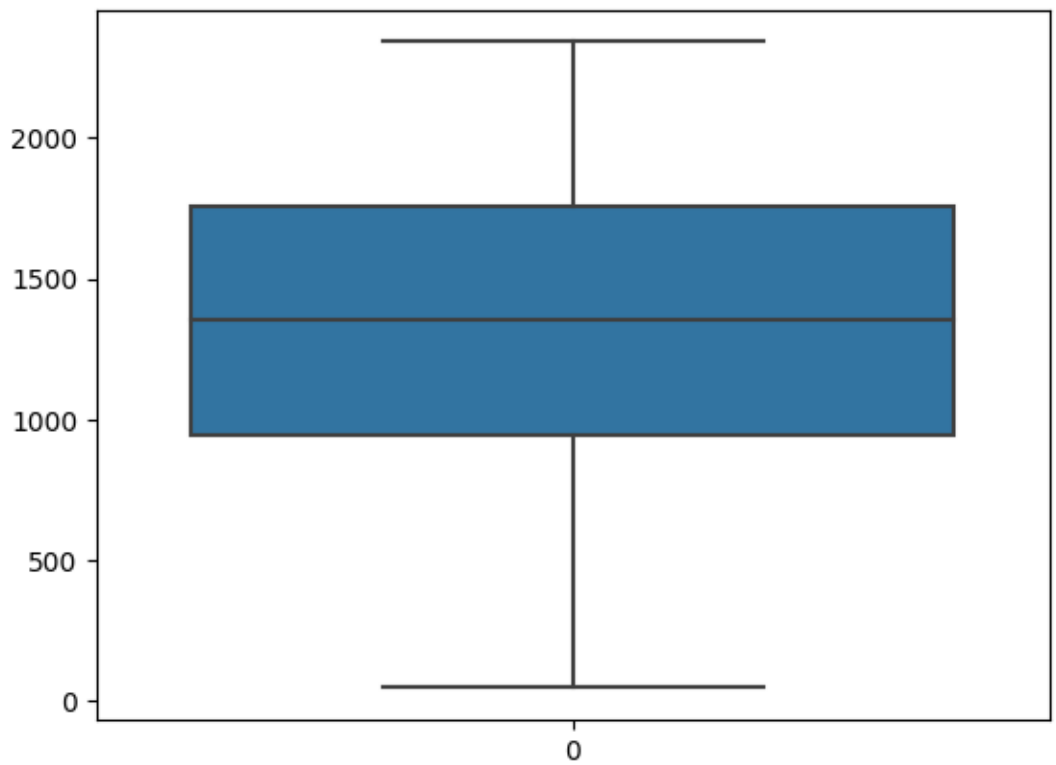


2. Replace

27 Replace the outliers by the maximum and minimum limit

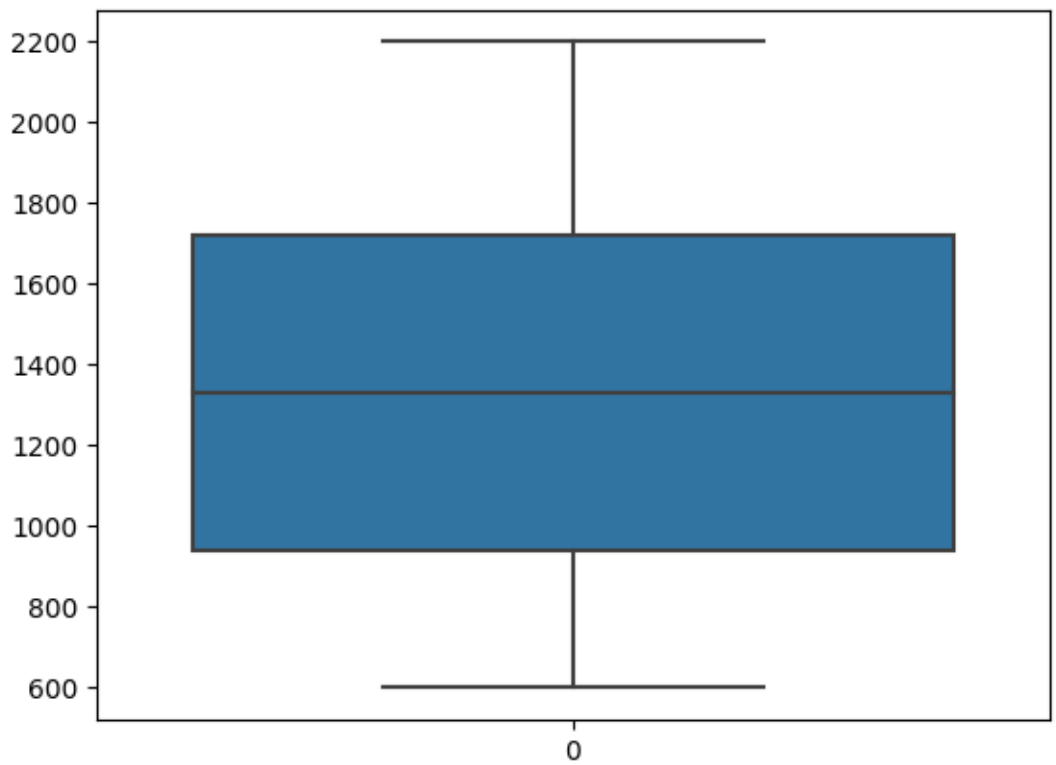
[22]:
```
project['project_replaced1'] = pd.DataFrame(np.
↪where(project['Actual_Shipment_Time'] > upper_limit1, upper_limit1, np.
↪where(project['Actual_Shipment_Time'] < lower_limit1, lower_limit1,␣
↪project['Actual_Shipment_Time'])))
sns.boxplot(project.project_replaced1)
```

[22]: <Axes: >



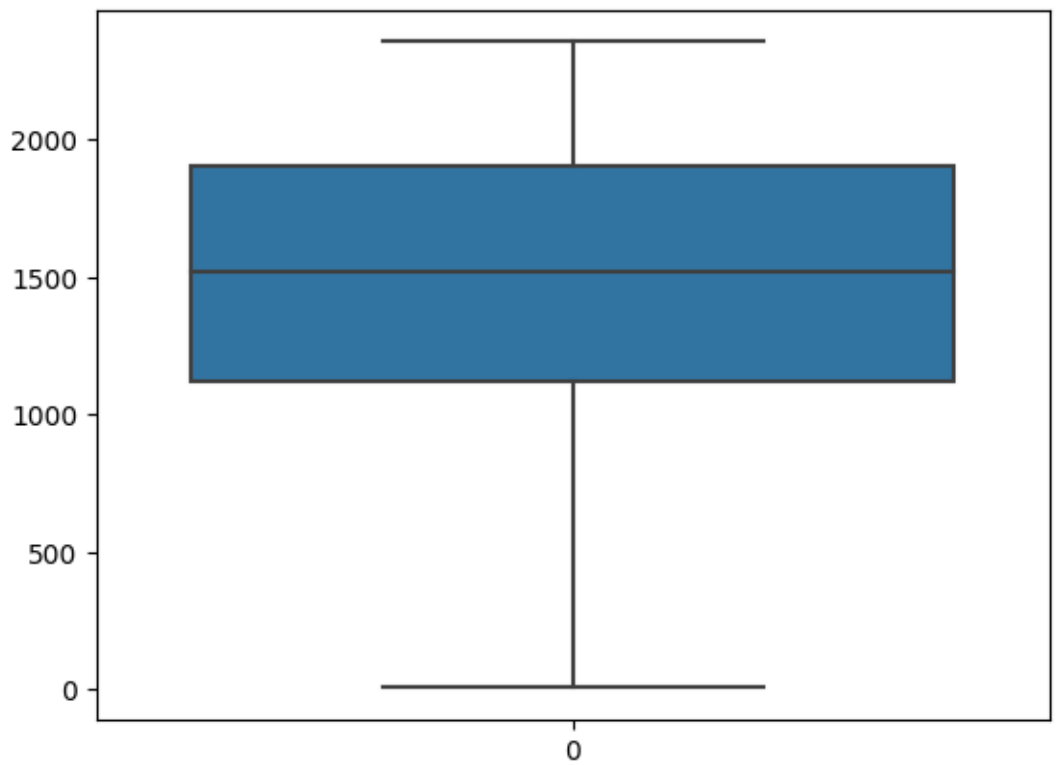
[23]:
```
project['project_replaced2'] = pd.DataFrame(np.
↪where(project['Planned_Shipment_Time'] > upper_limit2, upper_limit2, np.
↪where(project['Planned_Shipment_Time'] < lower_limit2, lower_limit2,␣
↪project['Planned_Shipment_Time'])))
sns.boxplot(project.project_replaced2)
```

[23]: <Axes: >



[24]:
```
project['project_replaced3'] = pd.DataFrame(np.
↪where(project['Planned_Delivery_Time'] > upper_limit1, upper_limit1, np.
↪where(project['Planned_Delivery_Time'] < lower_limit1, lower_limit1,␣
↪project['Planned_Delivery_Time'])))
sns.boxplot(project.project_replaced3)
```

[24]: <Axes: >



Dummy Variables

[25]:
```
import pandas as pd
import numpy as np
```

[26]:
```
project = pd.read_csv(r"/content/Datasets.csv")
```

[27]:
```
project.columns
```

[27]:
```
Index(['Year', 'Month', 'DayofMonth', 'DayOfWeek', 'Actual_Shipment_Time',
       'Planned_Shipment_Time', 'Planned_Delivery_Time', 'Carrier_Name',
       'Carrier_Num', 'Planned_TimeofTravel', 'Shipment_Delay', 'Source',
       'Destination', 'Distance', 'Delivery_Status'],
      dtype='object')
```

[28]:
```
project.shape
```

[28]:
```
(7999, 15)
```

[29]:
```
project.dtypes
```

[29]:
```
Year                       int64
Month                      int64
DayofMonth                 int64
DayOfWeek                  int64
Actual_Shipment_Time     float64
Planned_Shipment_Time      int64
Planned_Delivery_Time      int64
Carrier_Name              object
Carrier_Num                int64
Planned_TimeofTravel       int64
Shipment_Delay           float64
Source                    object
Destination               object
Distance                   int64
Delivery_Status          float64
dtype: object
```

[30]:
```python
project.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7999 entries, 0 to 7998
Data columns (total 15 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Year                   7999 non-null   int64
 1   Month                  7999 non-null   int64
 2   DayofMonth             7999 non-null   int64
 3   DayOfWeek              7999 non-null   int64
 4   Actual_Shipment_Time   7860 non-null   float64
 5   Planned_Shipment_Time  7999 non-null   int64
 6   Planned_Delivery_Time  7999 non-null   int64
 7   Carrier_Name           7999 non-null   object
 8   Carrier_Num            7999 non-null   int64
 9   Planned_TimeofTravel   7999 non-null   int64
 10  Shipment_Delay         7860 non-null   float64
 11  Source                 7999 non-null   object
 12  Destination            7999 non-null   object
 13  Distance               7999 non-null   int64
 14  Delivery_Status        7860 non-null   float64
dtypes: float64(3), int64(9), object(3)
memory usage: 937.5+ KB
```

[31]:
```python
# Drop Actual_Shipment_Time column

project1 = project.drop(['Carrier_Name' , 'Planned_Shipment_Time' ,␣
 ↪'Planned_Delivery_Time'], axis = 1)
```

```
project.drop(['Carrier_Name' , 'Planned_Shipment_Time' ,␣
↪'Planned_Delivery_Time'], axis = 1, inplace = True)
```

[32]:
```
# Create dummy variables
project_new = pd.get_dummies(project)
project_new_1 = pd.get_dummies(project, drop_first = True)
```

[34]:
```
# Created dummies for all categorical columns
##### One Hot Encoding works
project.columns
project = project[['Year', 'Month', 'DayofMonth', 'DayOfWeek',␣
↪'Actual_Shipment_Time',
          'Carrier_Num', 'Planned_TimeofTravel', 'Shipment_Delay', 'Source',
       'Destination', 'Distance', 'Delivery_Status']]
```

[35]:
```
a = project['DayofMonth']
b = project[['DayofMonth']]
```

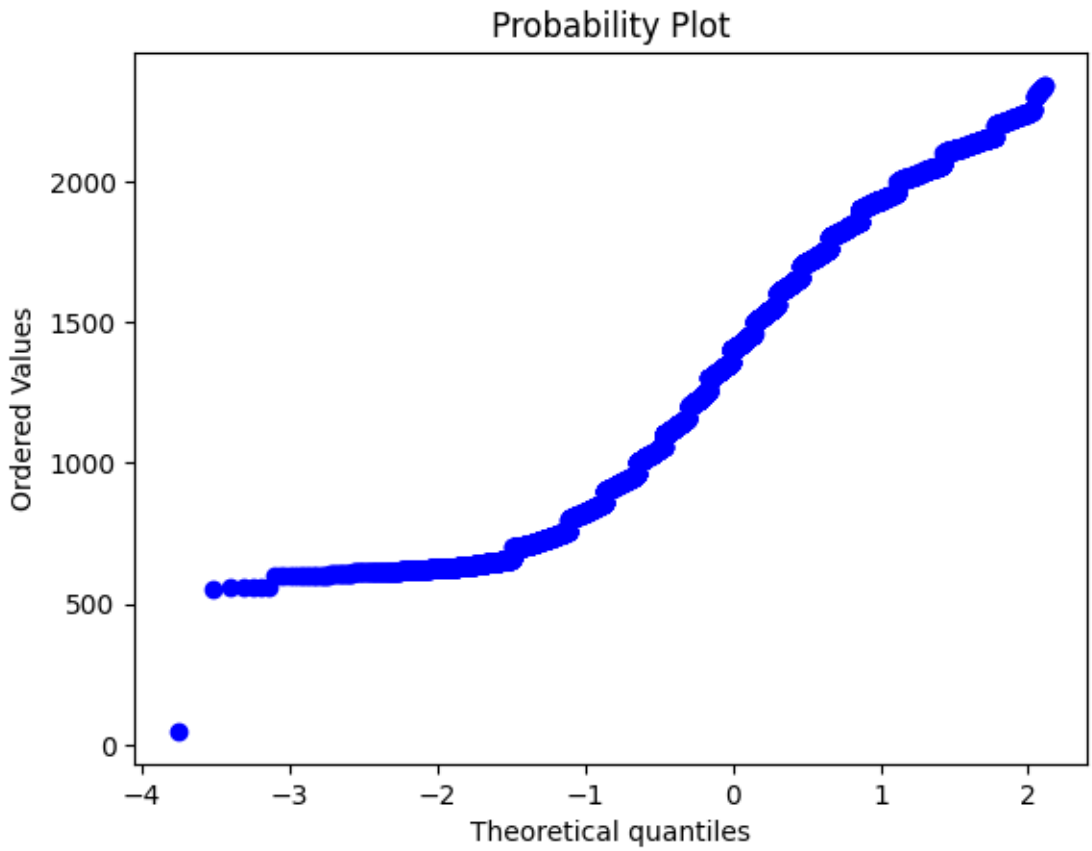
Transformation

[36]:
```
import pandas as pd
import numpy as np
import scipy.stats as stats
import pylab
```

[37]:
```
project = pd.read_csv(r"/content/Datasets.csv")
```

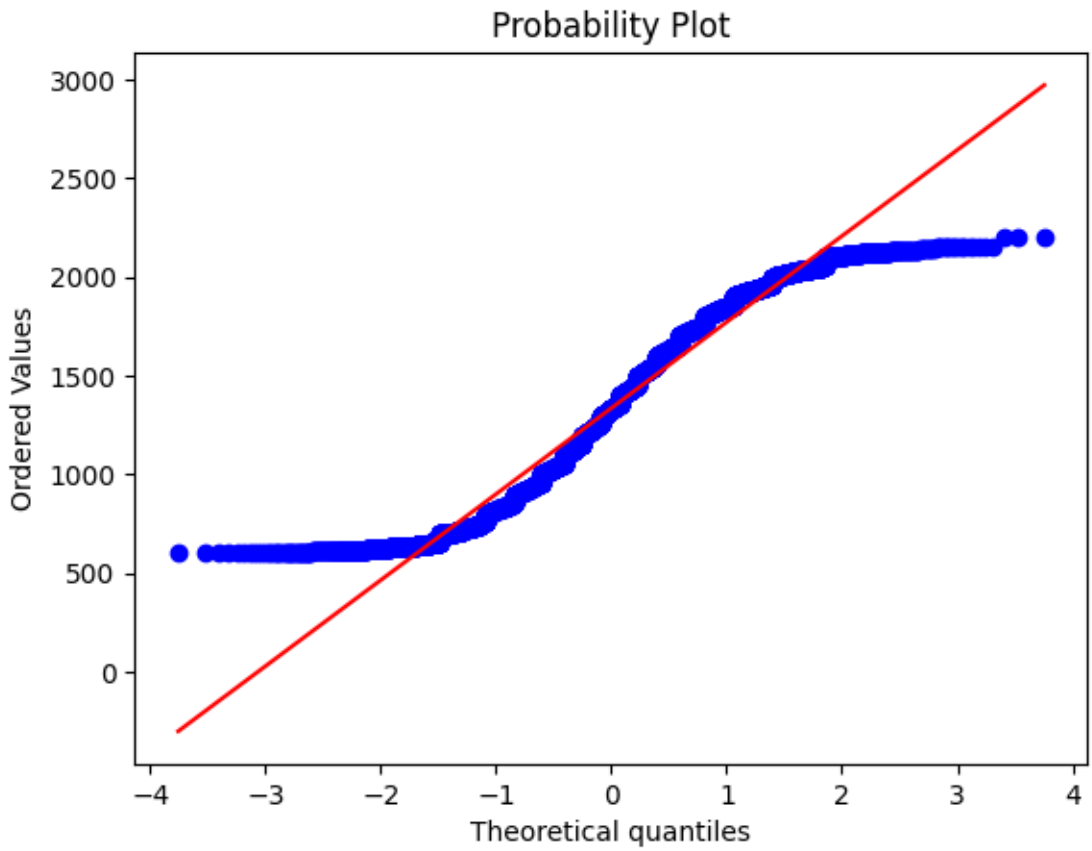
[38]:
```
#  normally distributed
stats.probplot(project.Actual_Shipment_Time, dist = "norm", plot = pylab)
```

[38]:
```
((array([-3.75505857, -3.52677228, -3.40129331, …,  3.40129331,
          3.52677228,  3.75505857]),
  array([ 47., 555., 558., …,  nan,  nan,  nan])),
 (nan, nan, nan))
```



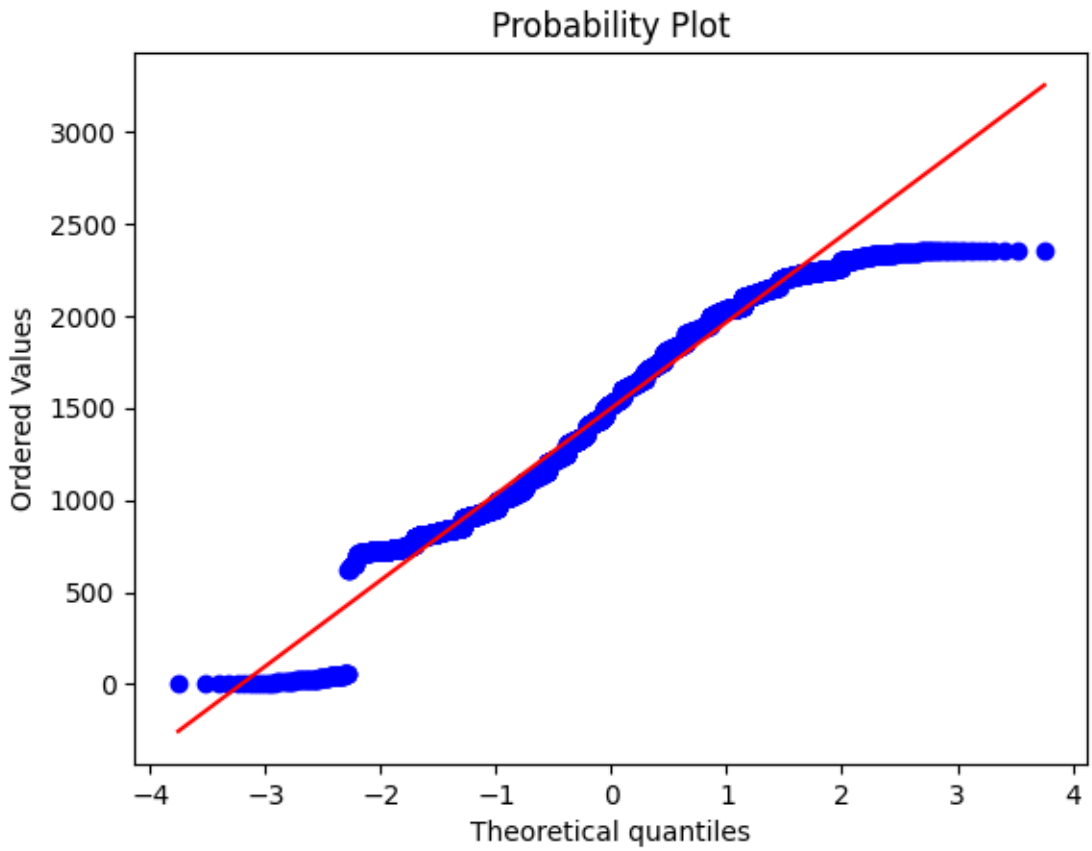
[39]:
```
stats.probplot(project.Planned_Shipment_Time, dist = "norm", plot = pylab)
```

[39]:
```
((array([-3.75505857, -3.52677228, -3.40129331, …,  3.40129331,
          3.52677228,  3.75505857]),
  array([ 600,  600,  600, …, 2200, 2200, 2200])),
 (435.96579572678934, 1335.3175396924617, 0.9768036704470942))
```



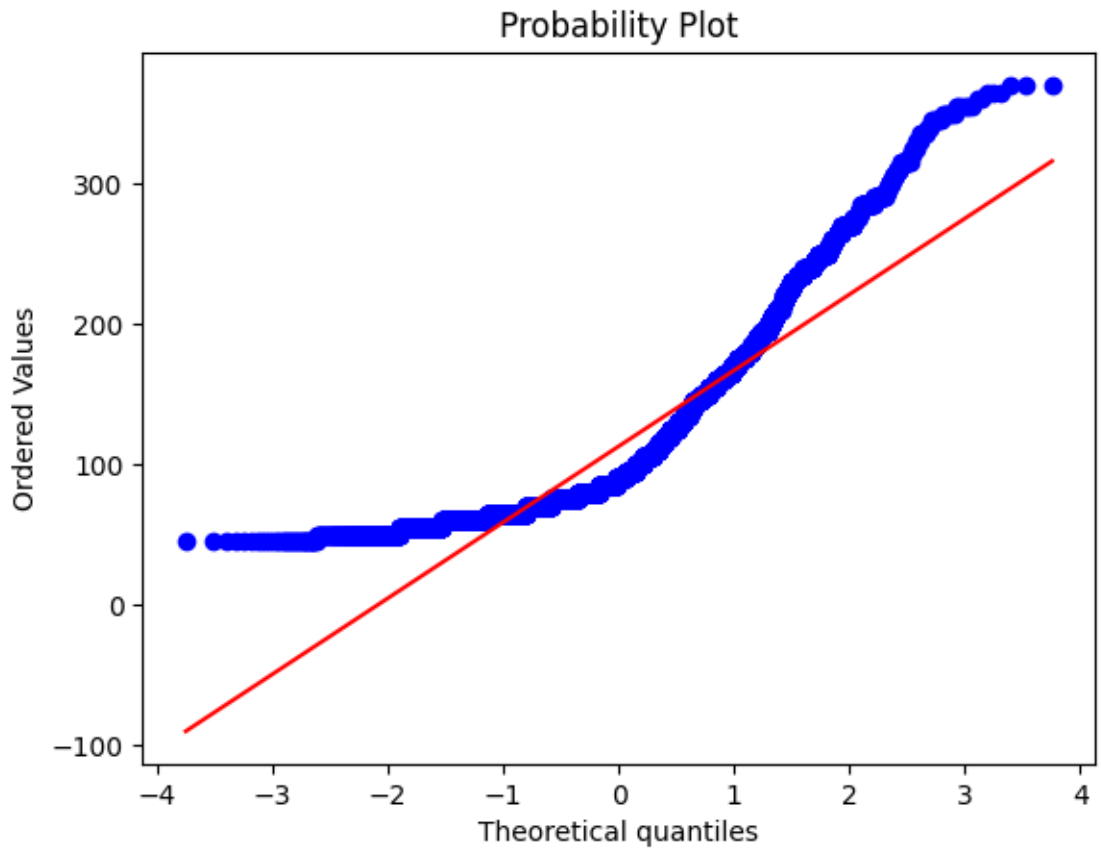
[42]:
```
stats.probplot(project.Planned_Delivery_Time, dist = "norm", plot = pylab)
```

[42]:
```
((array([-3.75505857, -3.52677228, -3.40129331, …,  3.40129331,
          3.52677228,  3.75505857]),
  array([   5,    5,    5, …, 2355, 2355, 2355])),
 (466.9271375575307, 1498.2554069258658, 0.9851475869914547))
```



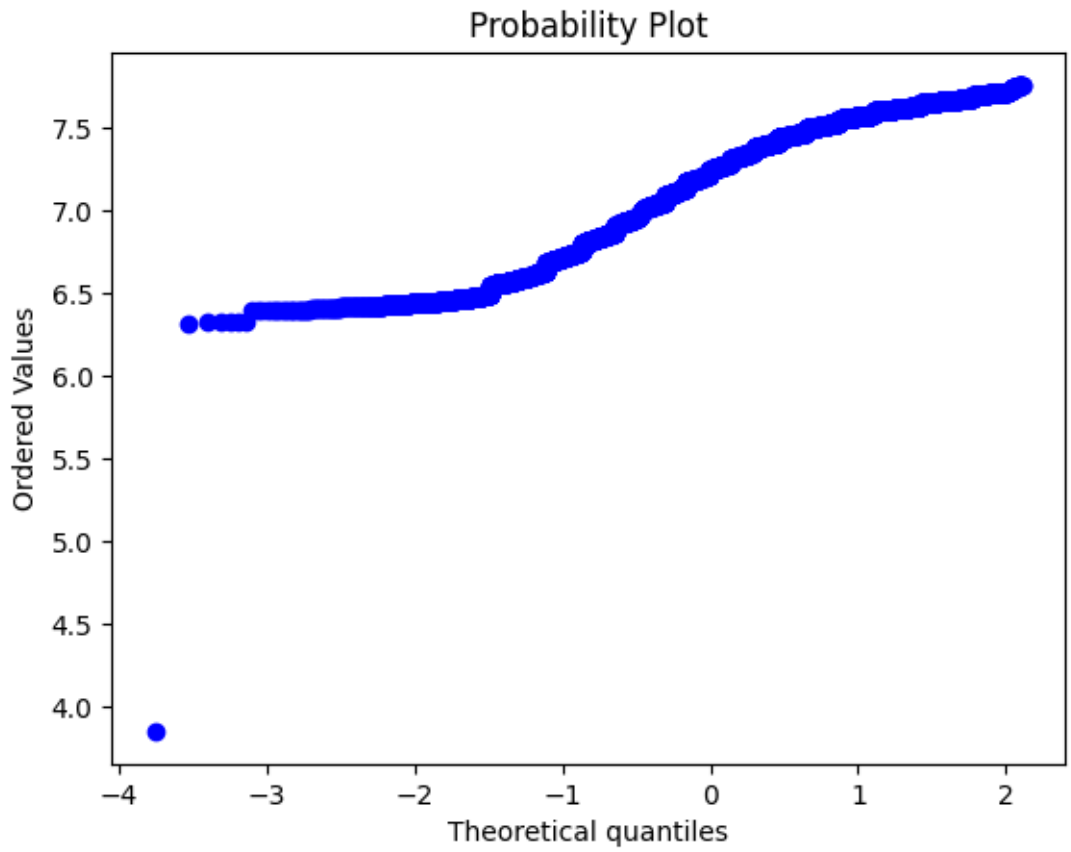
[41]:
```
stats.probplot(project.Planned_TimeofTravel, dist = "norm", plot = pylab)
```

[41]:
```
((array([-3.75505857, -3.52677228, -3.40129331, …,  3.40129331,
          3.52677228,  3.75505857]),
  array([ 45,  45,  45, …, 370, 370, 370])),
 (54.05385289105414, 112.89911238904863, 0.9194687259293179))
```



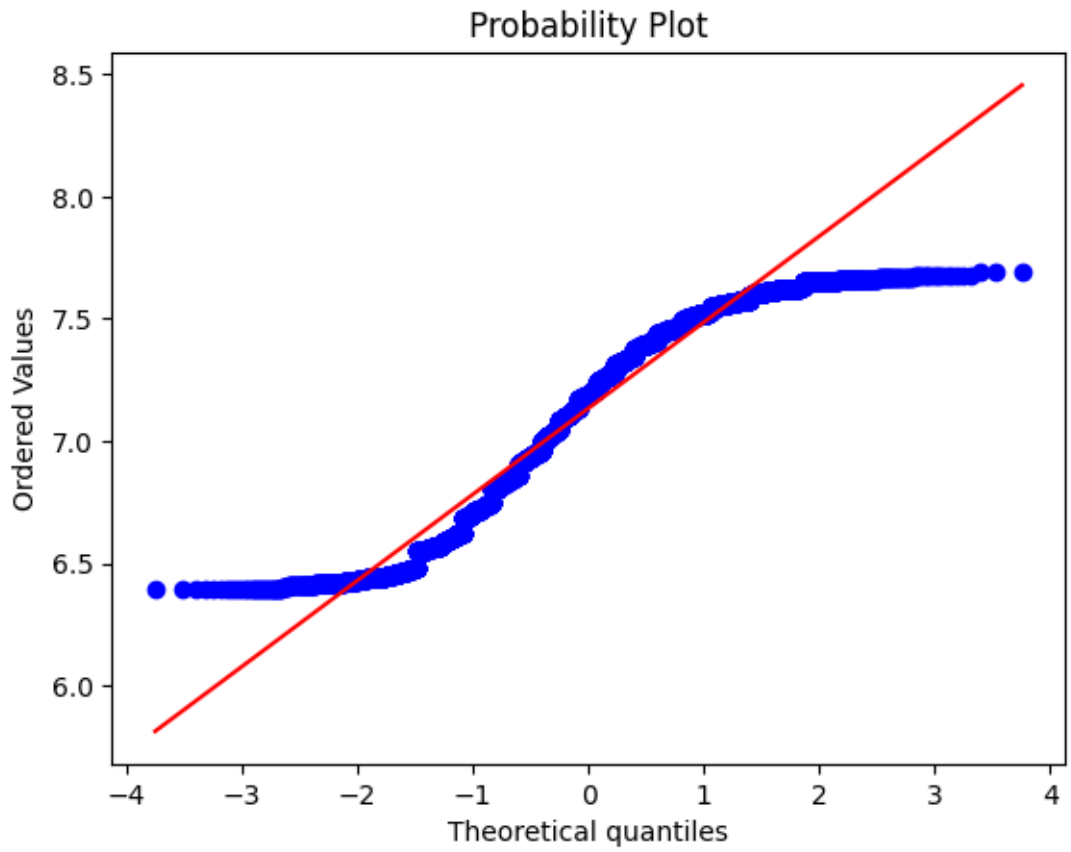
[46]:
```
# log Transformation

stats.probplot (np.log(project.Actual_Shipment_Time), dist = "norm", plot =␣
↪pylab)
```

[46]:
```
((array([-3.75505857, -3.52677228, -3.40129331, …,  3.40129331,
         3.52677228,  3.75505857]),
  array([3.8501476 , 6.31896811, 6.32435896, …,        nan,        nan,
                nan])),
 (nan, nan, nan))
```



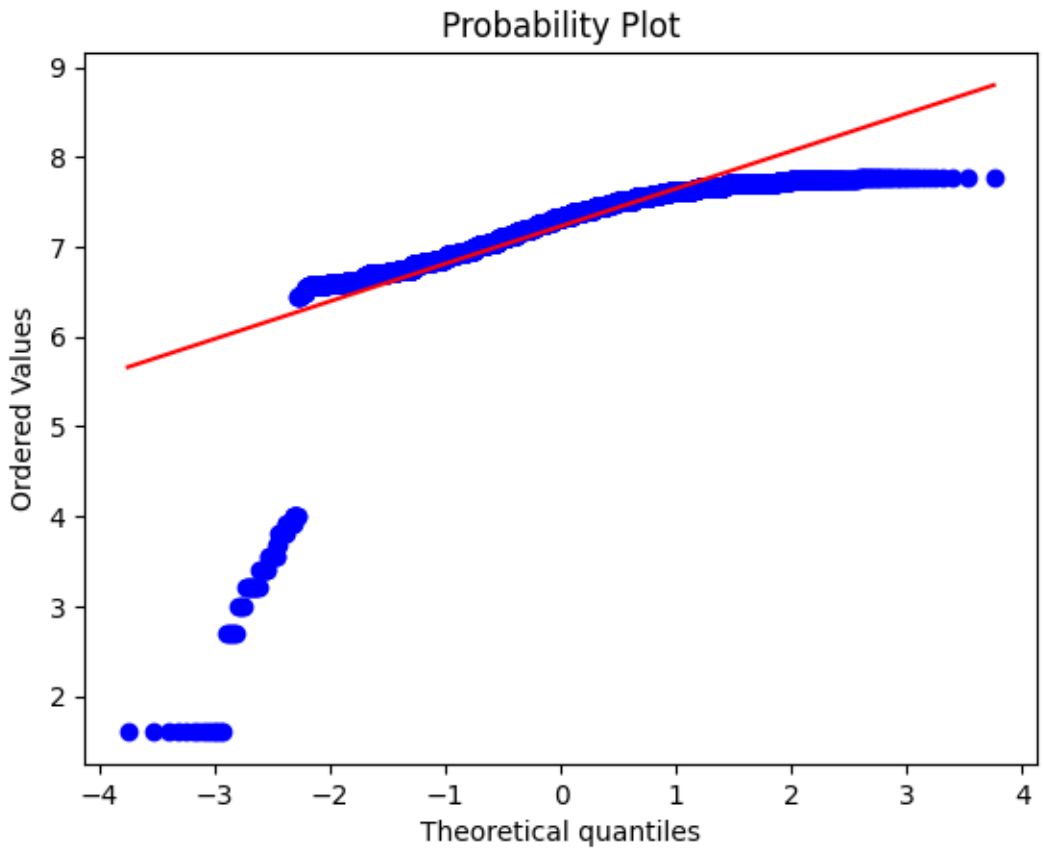
[45]:
```
stats.probplot (np.log(project.Planned_Shipment_Time), dist = "norm", plot =␣
↪pylab)
```

[45]:
```
((array([-3.75505857, -3.52677228, -3.40129331, …,  3.40129331,
          3.52677228,  3.75505857]),
  array([6.39692966, 6.39692966, 6.39692966, …, 7.69621264, 7.69621264,
         7.69621264])),
 (0.3516247486607927, 7.1351139383757465, 0.9706500937589305))
```



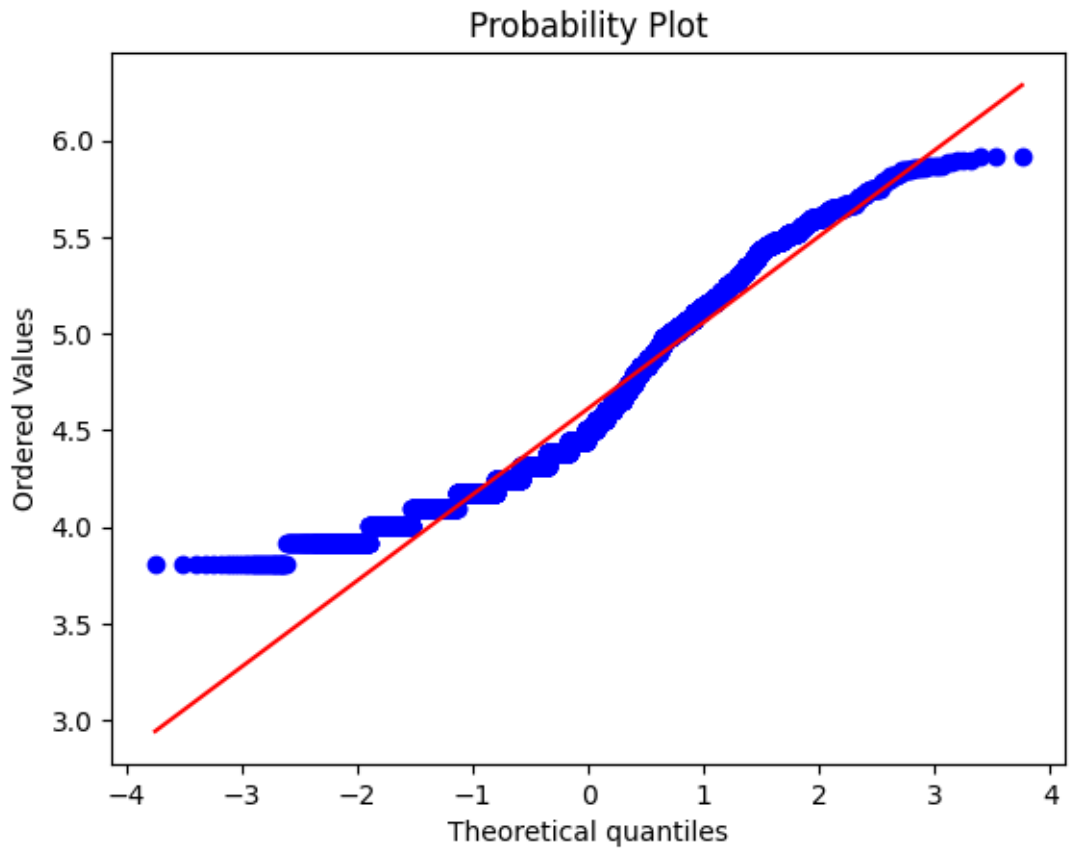
[47]:
```
stats.probplot(np.log(project.Planned_Delivery_Time), dist = "norm", plot =␣
↪pylab)
```

[47]:
```
((array([-3.75505857, -3.52677228, -3.40129331, …,  3.40129331,
          3.52677228,  3.75505857]),
 array([1.60943791, 1.60943791, 1.60943791, …, 7.76429601, 7.76429601,
        7.76429601])),
 (0.41791891269380516, 7.2296711133439935, 0.7793248784604291))
```



[48]:
```
stats.probplot(np.log(project.Planned_TimeofTravel), dist = "norm", plot =␣
↪pylab)
```

[48]:
```
((array([-3.75505857, -3.52677228, -3.40129331, …,  3.40129331,
          3.52677228,  3.75505857]),
 array([3.80666249, 3.80666249, 3.80666249, …, 5.91350301, 5.91350301,
        5.91350301])),
 (0.4452819531294663, 4.613987512354753, 0.973204139040742))
```



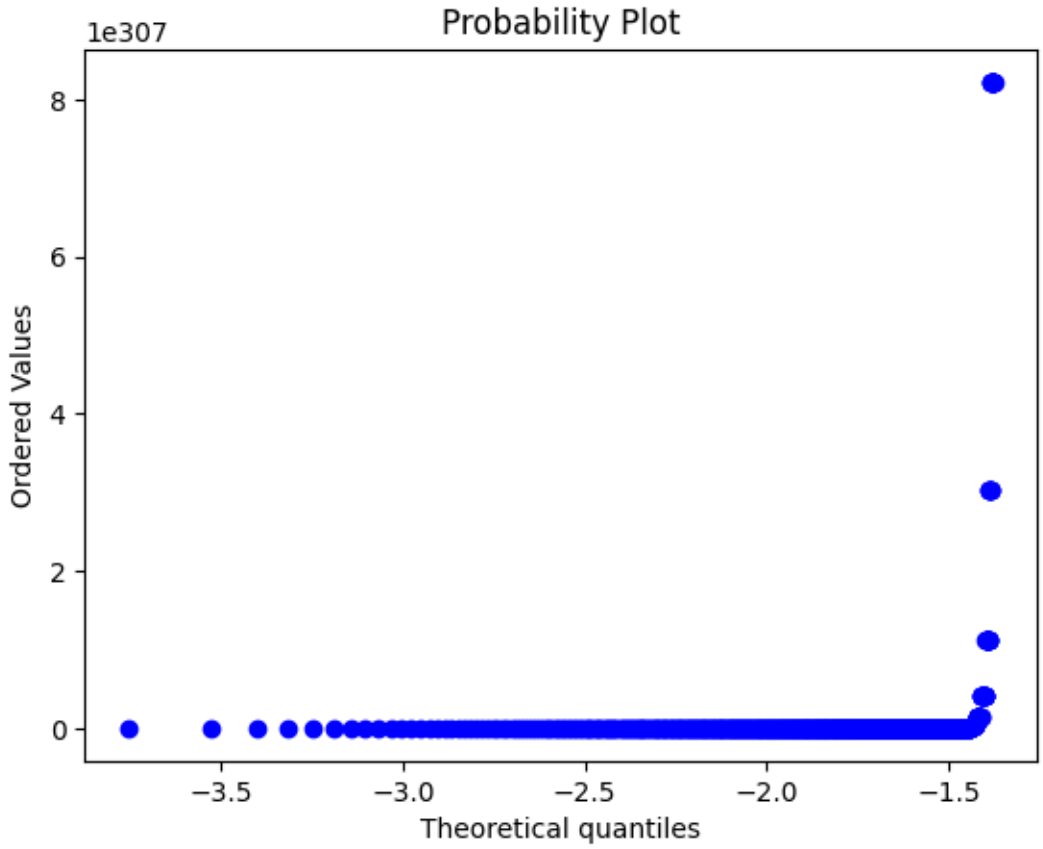
[49]:
```
# exp Transformation

stats.probplot (np.exp(project.Actual_Shipment_Time), dist = "norm", plot =␣
↪pylab)
```

```
/usr/local/lib/python3.10/dist-packages/pandas/core/arraylike.py:402:
RuntimeWarning: overflow encountered in exp
  result = getattr(ufunc, method)(*inputs, **kwargs)
/usr/local/lib/python3.10/dist-packages/numpy/core/_methods.py:180:
RuntimeWarning: overflow encountered in reduce
  ret = umr_sum(arr, axis, dtype, out, keepdims, where=where)
```

[49]:
```
((array([-3.75505857, -3.52677228, -3.40129331, …,  3.40129331,
         3.52677228,  3.75505857]),
 array([2.58131289e+020, 1.08003407e+241, 2.16930642e+242, …,
                    nan,             nan,             nan])),
 (nan, nan, nan))
```

```
/usr/local/lib/python3.10/dist-packages/matplotlib/ticker.py:2094:
RuntimeWarning: overflow encountered in multiply
```

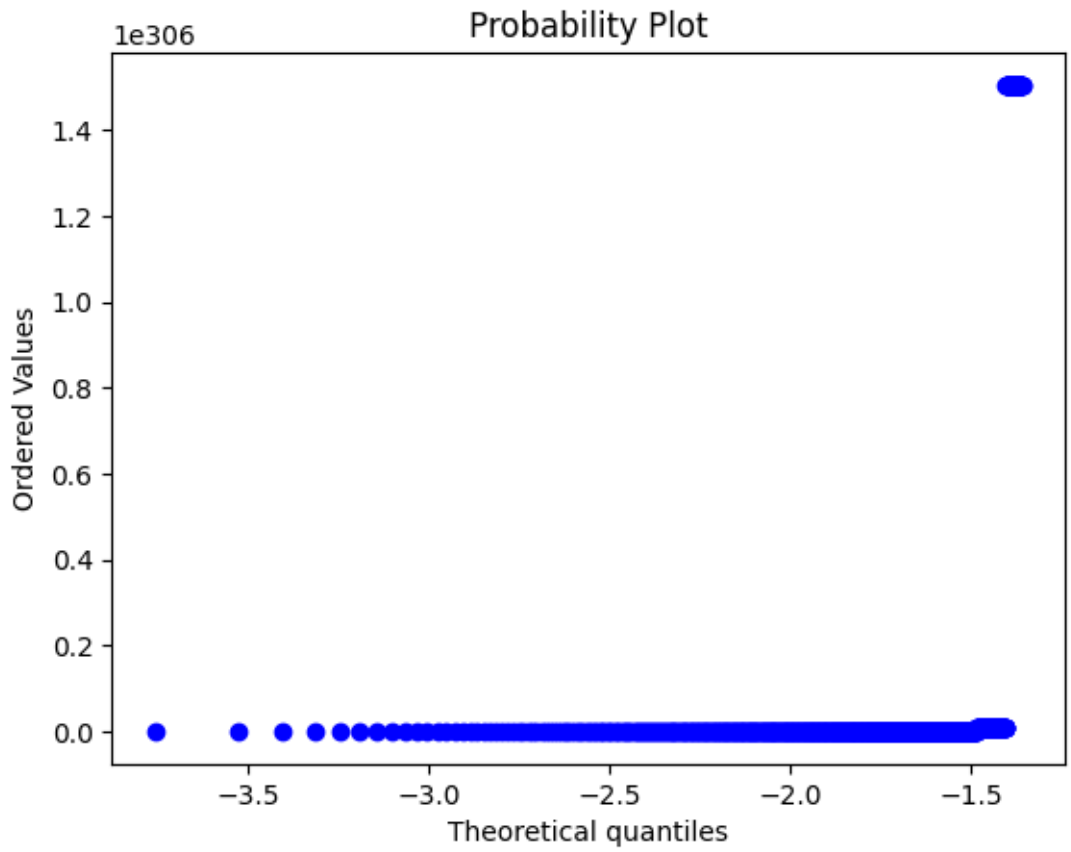
```
steps = self._extended_steps * scale
```



[50]:
```
stats.probplot (np.exp(project.Planned_Shipment_Time), dist = "norm", plot =␣
↪pylab)
```

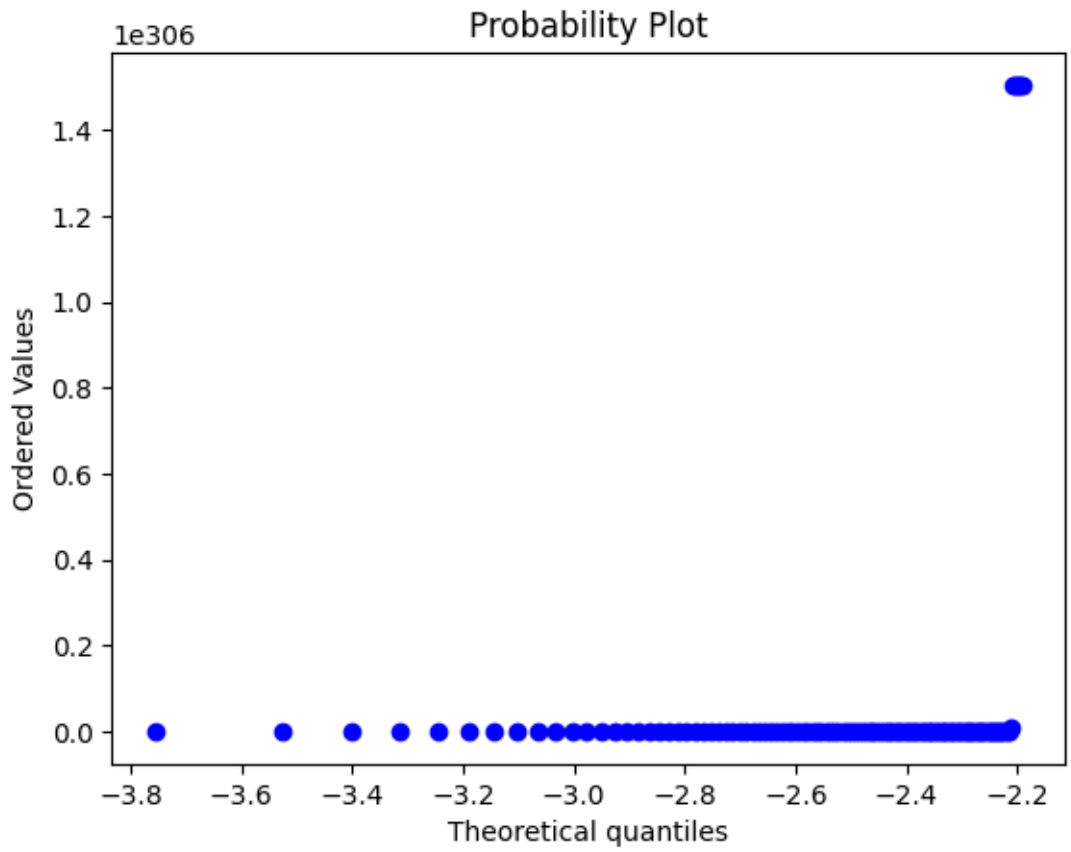
```
/usr/local/lib/python3.10/dist-packages/numpy/lib/function_base.py:2698:
RuntimeWarning: invalid value encountered in subtract
  X -= avg[:, None]
```

[50]:
```
((array([-3.75505857, -3.52677228, -3.40129331, …,  3.40129331,
          3.52677228,  3.75505857]),
 array([3.7730203e+260, 3.7730203e+260, 3.7730203e+260, …,
                   inf,            inf,            inf])),
 (nan, nan, nan))
```



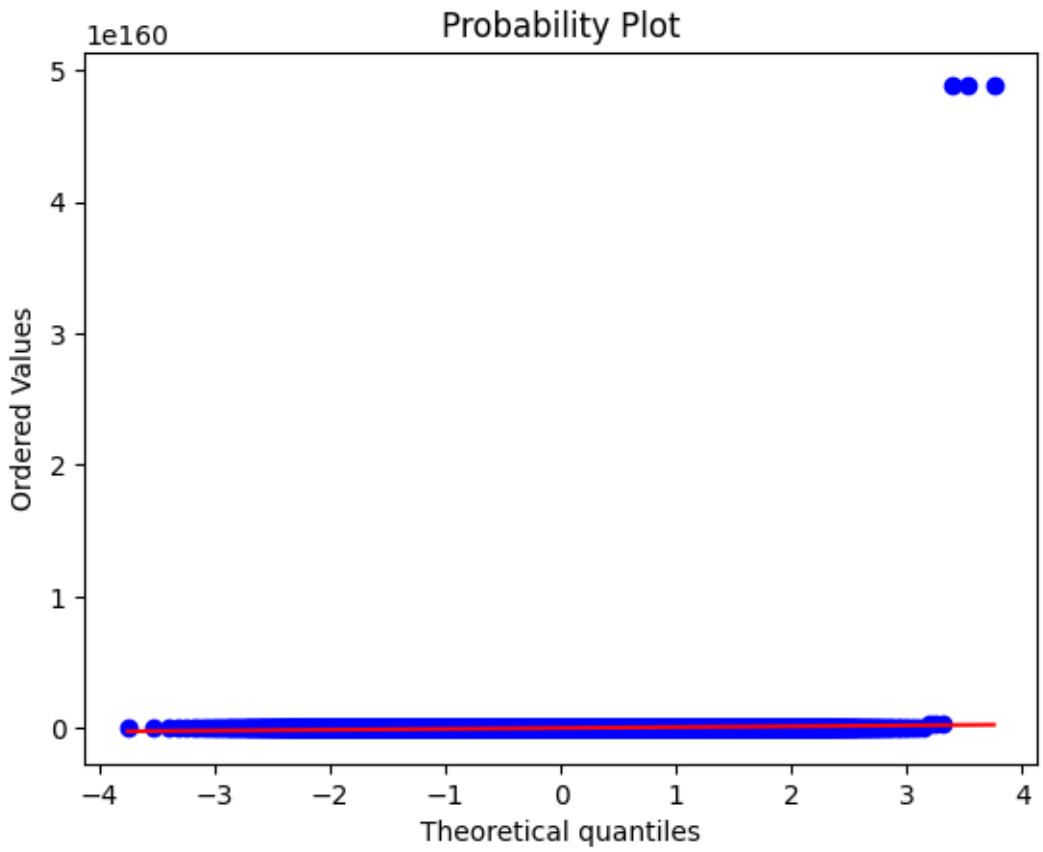
[51]:
```
stats.probplot(np.exp(project.Planned_Delivery_Time), dist = "norm", plot =␣
↪pylab)
```

[51]:
```
((array([-3.75505857, -3.52677228, -3.40129331, …,  3.40129331,
          3.52677228,  3.75505857]),
 array([148.4131591, 148.4131591, 148.4131591, …,         inf,
                inf,         inf])),
 (nan, nan, nan))
```



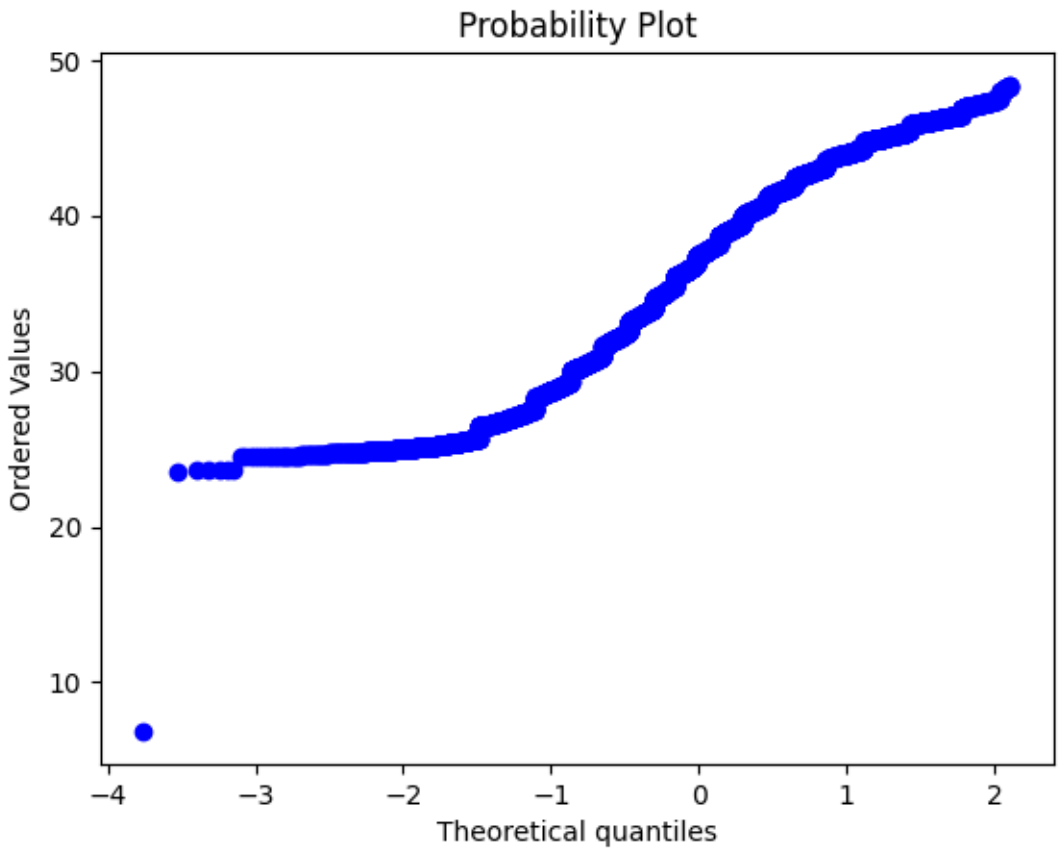
[52]:
```
stats.probplot(np.exp(project.Planned_TimeofTravel), dist = "norm", plot =␣
↪pylab)
```

[52]:
```
((array([-3.75505857, -3.52677228, -3.40129331, …,  3.40129331,
          3.52677228,  3.75505857]),
  array([3.49342711e+019, 3.49342711e+019, 3.49342711e+019, …,
         4.88605447e+160, 4.88605447e+160, 4.88605447e+160])),
 (6.571660879246802e+157, 1.844903365428962e+157, 0.0))
```



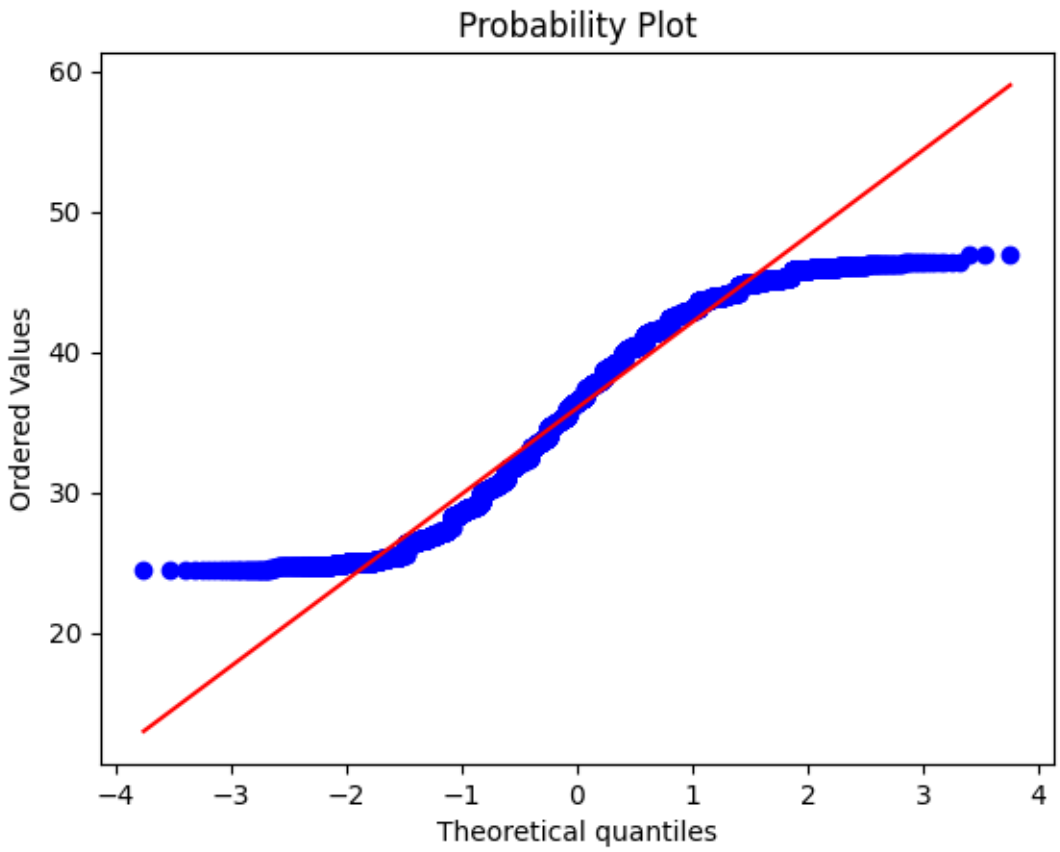
[53]:
```
# sqrt Transformation

stats.probplot (np.sqrt(project.Actual_Shipment_Time), dist = "norm", plot =␣
↪pylab)
```

[53]:
```
((array([-3.75505857, -3.52677228, -3.40129331, …,  3.40129331,
         3.52677228,  3.75505857]),
  array([ 6.8556546 , 23.55843798, 23.62202362, …,         nan,
                nan,         nan])),
 (nan, nan, nan))
```



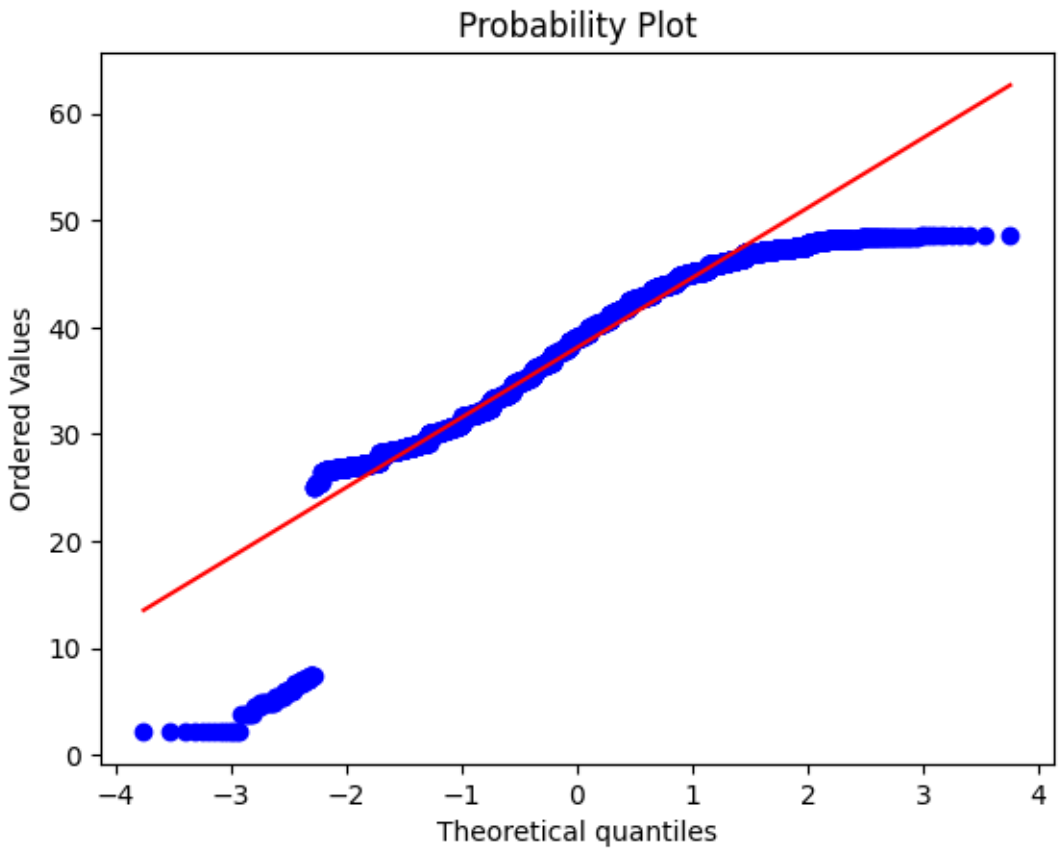
[54]:
```
stats.probplot (np.sqrt(project.Planned_Shipment_Time), dist = "norm", plot =␣
↪pylab)
```

[54]:
```
((array([-3.75505857, -3.52677228, -3.40129331, …,  3.40129331,
          3.52677228,  3.75505857]),
  array([24.49489743, 24.49489743, 24.49489743, …, 46.9041576 ,
         46.9041576 , 46.9041576 ])),
 (6.1251327297262055, 35.999726417463044, 0.9761656641251333))
```



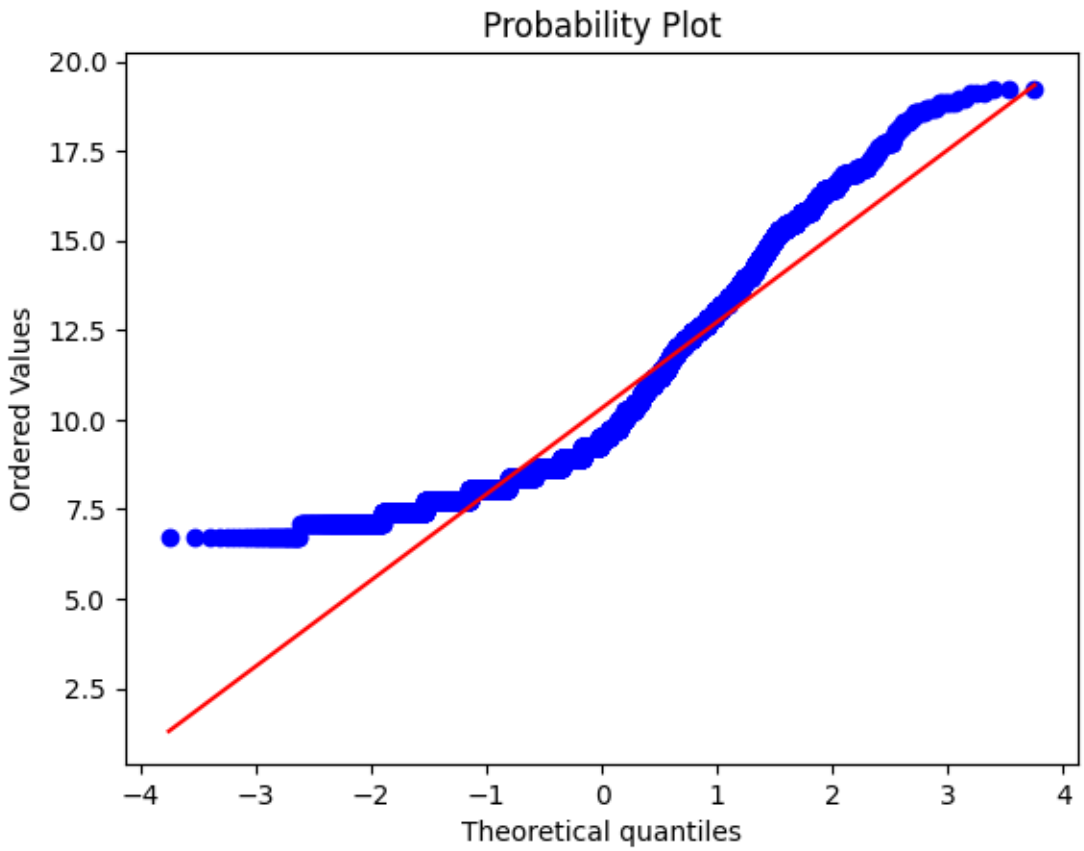
[55]:
```
stats.probplot(np.sqrt(project.Planned_Delivery_Time), dist = "norm", plot =␣
↪pylab)
```

[55]:
```
((array([-3.75505857, -3.52677228, -3.40129331, …,  3.40129331,
          3.52677228,  3.75505857]),
  array([ 2.23606798,  2.23606798,  2.23606798, …, 48.52834223,
         48.52834223, 48.52834223])),
 (6.535515605318326, 38.097194290880964, 0.954316430509373))
```



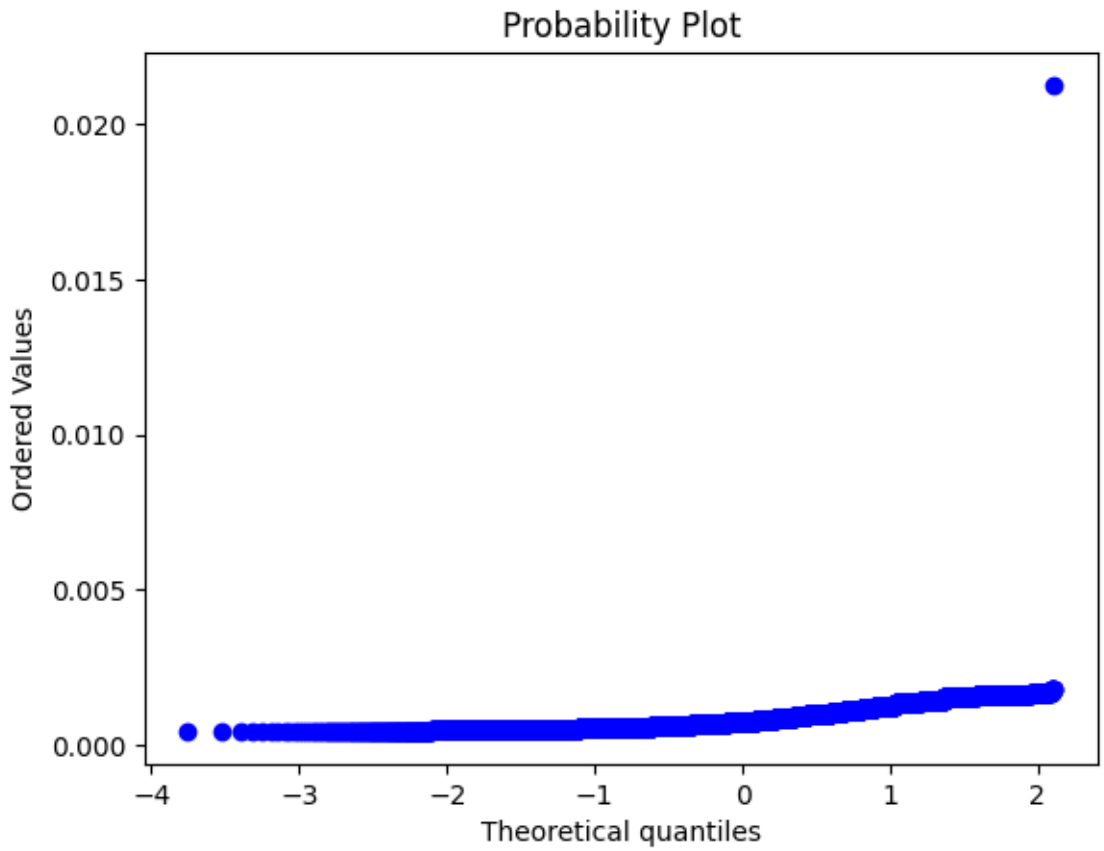
[56]:
```
stats.probplot(np.sqrt(project.Planned_TimeofTravel), dist = "norm", plot =␣
↪pylab)
```

[56]:
```
((array([-3.75505857, -3.52677228, -3.40129331, …,  3.40129331,
          3.52677228,  3.75505857]),
  array([ 6.70820393,  6.70820393,  6.70820393, …, 19.23538406,
         19.23538406, 19.23538406])),
 (2.4016185165026167, 10.321561127388817, 0.9515513514518762))
```



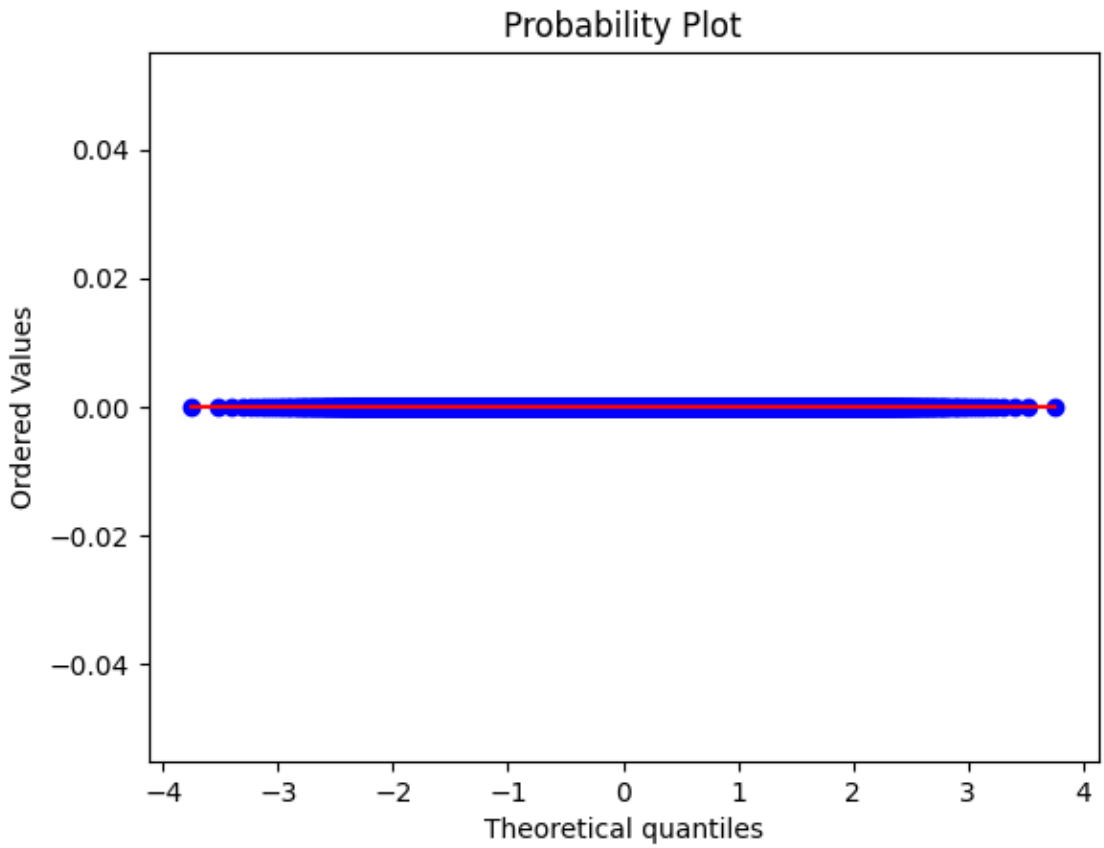
[57]:
```
# reciprocal Transformation

stats.probplot (np.reciprocal(project.Actual_Shipment_Time), dist = "norm",␣
↪plot = pylab)
```

[57]:
```
((array([-3.75505857, -3.52677228, -3.40129331, …,  3.40129331,
          3.52677228,  3.75505857]),
  array([0.00042717, 0.00042717, 0.00042845, …,        nan,        nan,
               nan])),
 (nan, nan, nan))
```



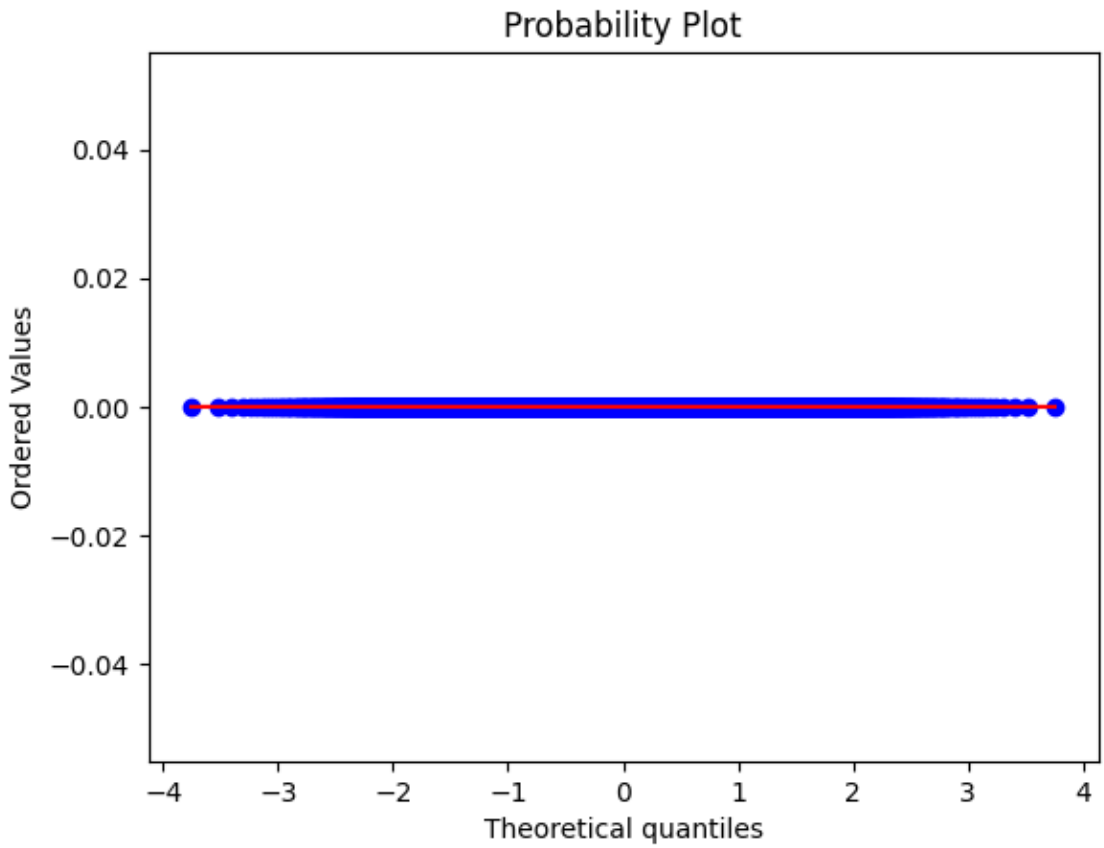
[58]:
```
stats.probplot (np.reciprocal(project.Planned_Shipment_Time), dist = "norm",␣
↪plot = pylab)
```

[58]:
```
((array([-3.75505857, -3.52677228, -3.40129331, …,  3.40129331,
          3.52677228,  3.75505857]),
 array([0, 0, 0, …, 0, 0, 0])),
(0.0, 0.0, 0.0))
```



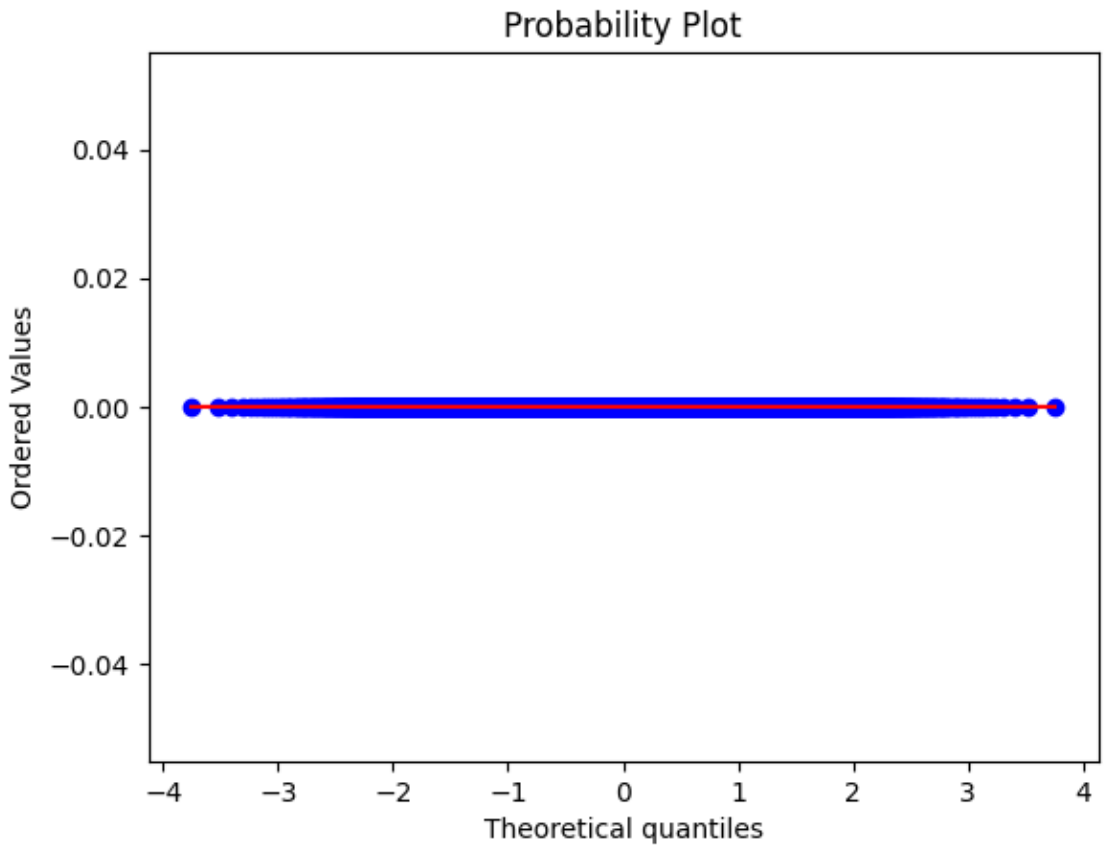
[59]:
```
stats.probplot(np.reciprocal(project.Planned_Delivery_Time), dist = "norm",␣
↪plot = pylab)
```

[59]:
```
((array([-3.75505857, -3.52677228, -3.40129331, …,  3.40129331,
          3.52677228,  3.75505857]),
  array([0, 0, 0, …, 0, 0, 0])),
 (0.0, 0.0, 0.0))
```



[60]:
```
stats.probplot(np.reciprocal(project.Planned_TimeofTravel), dist = "norm", plot␣
↪= pylab)
```

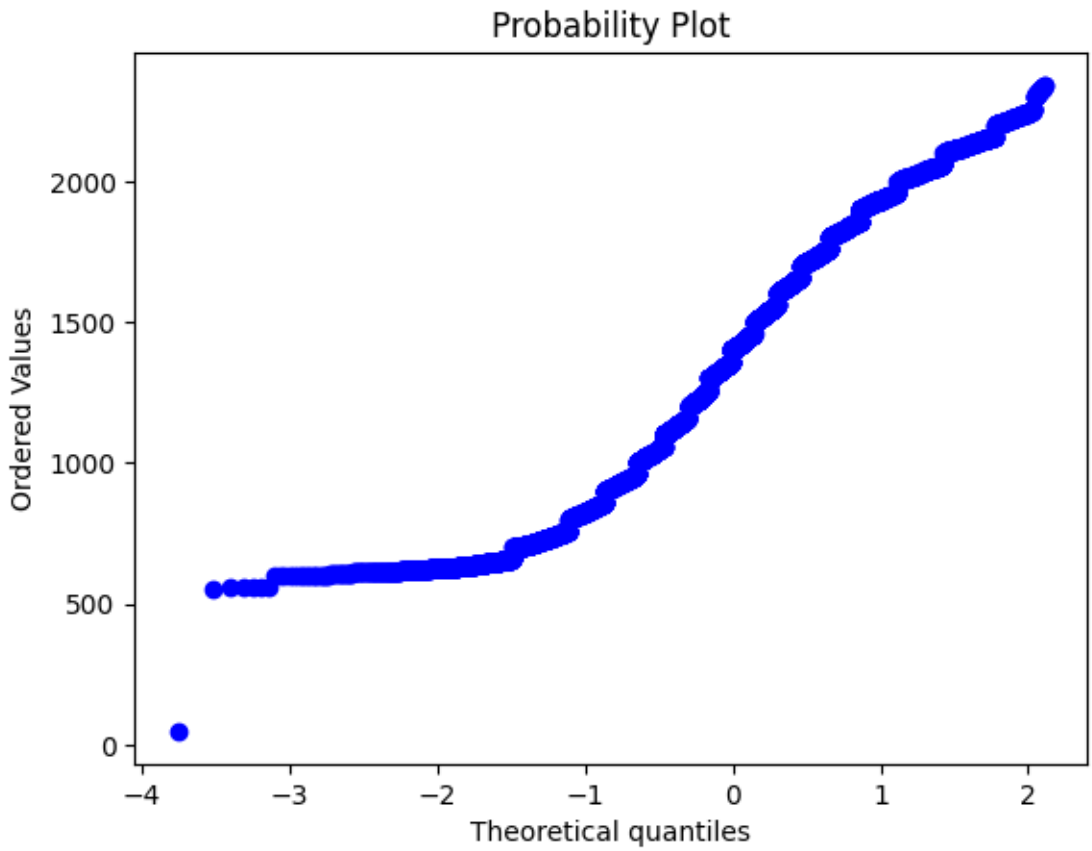
[60]:
```
((array([-3.75505857, -3.52677228, -3.40129331, …,  3.40129331,
          3.52677228,  3.75505857]),
  array([0, 0, 0, …, 0, 0, 0])),
 (0.0, 0.0, 0.0))
```



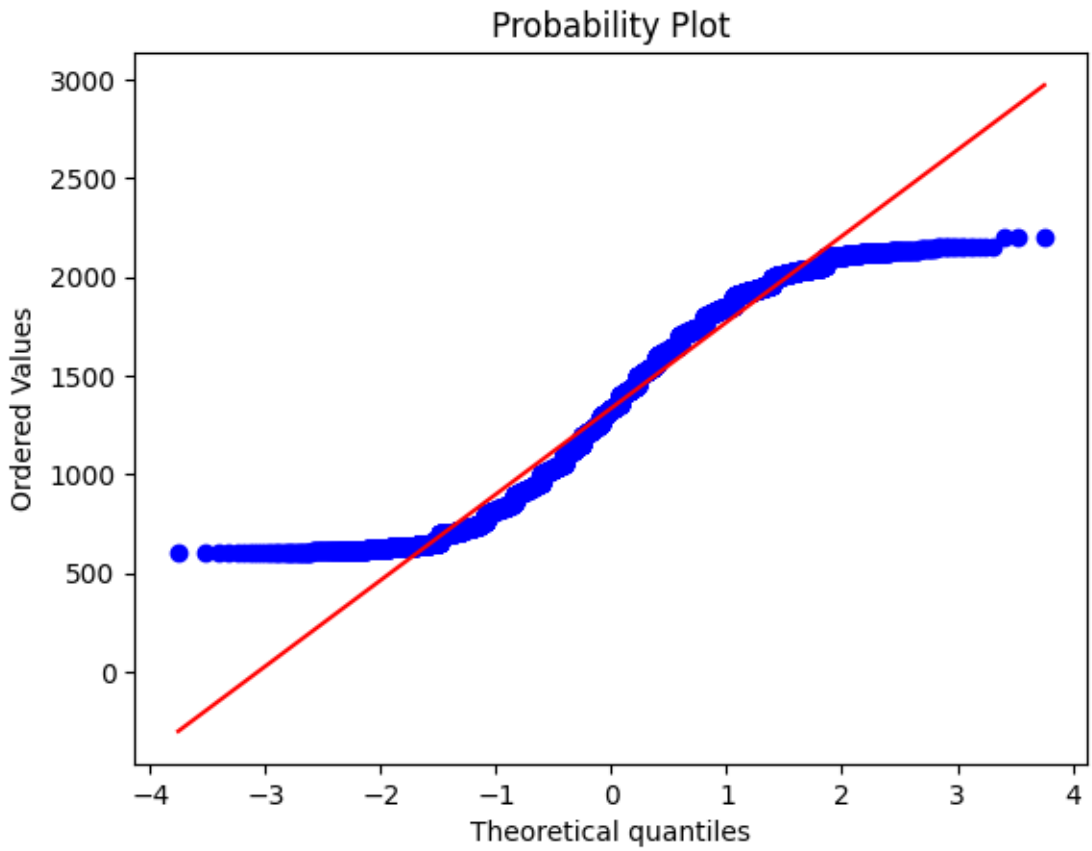
[61]:
```
# Box cox Transformation

import pandas as pd
from scipy import stats

# Plotting related modules
import seaborn as sns
import matplotlib.pyplot as plt
import pylab
```

[62]:
```
project = pd.read_csv(r"/content/Datasets.csv")
```

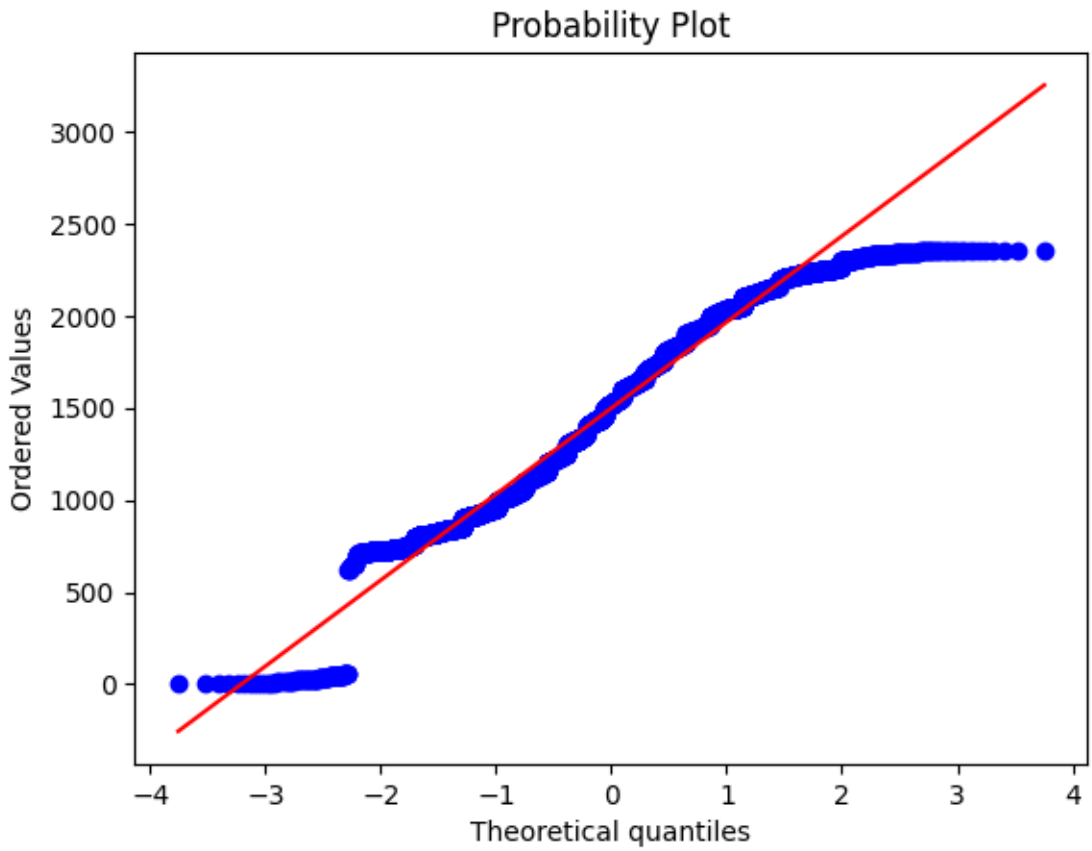
[63]:
```
# Original data
prob1 = stats.probplot(project.Actual_Shipment_Time, dist = stats.norm, plot =␣
↪pylab)
```



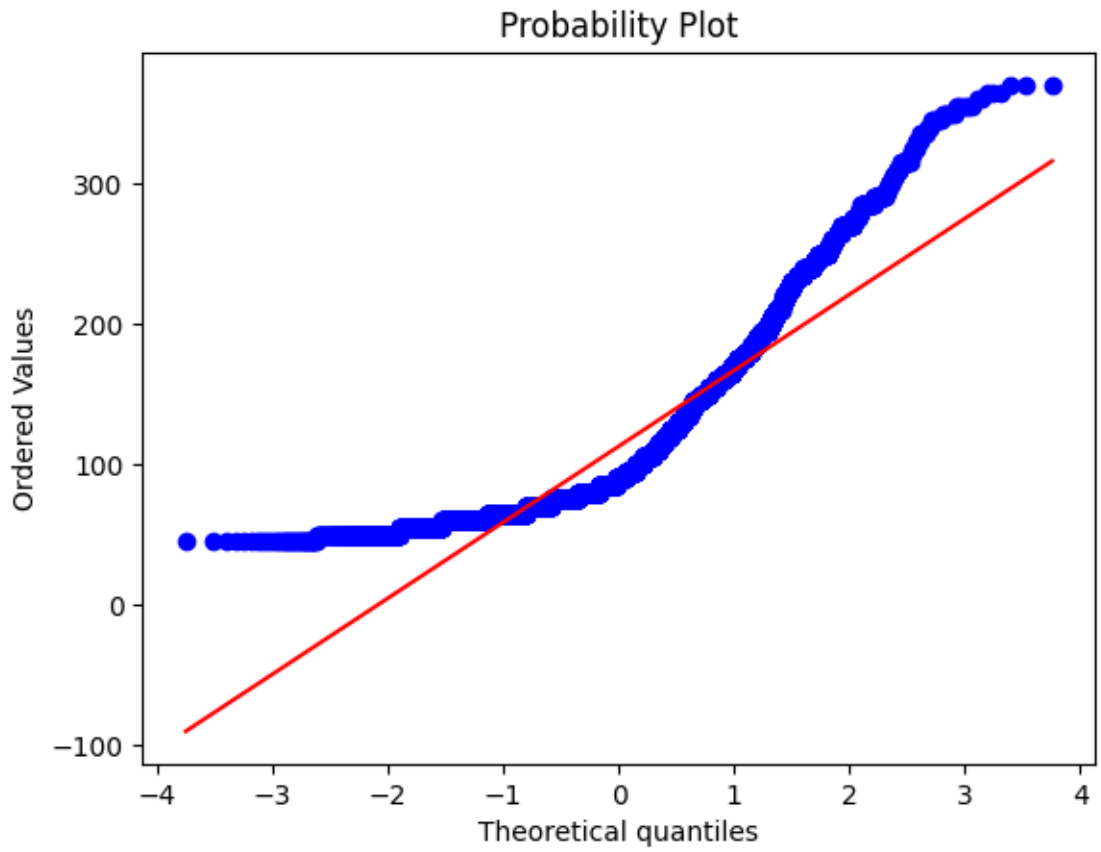
[64]:
```
prob2 = stats.probplot(project.Planned_Shipment_Time, dist = stats.norm, plot =␣
↪pylab)
```



[65]:
```
prob3 = stats.probplot(project.Planned_Delivery_Time, dist = stats.norm, plot =␣
↪pylab)
```



[66]:
```
prob4= stats.probplot(project.Planned_TimeofTravel, dist = stats.norm, plot =
 ↪pylab)
```



28 Transform training data & save lambda value

[78]:
```
fitted_data1, fitted_lambda1 = stats.boxcox(project.Actual_Shipment_Time)
```

```
---------------------------------------------------------------------------
BracketError                              Traceback (most recent call last)
<ipython-input-78-b0e564be35d7> in <cell line: 1>()
----> 1 fitted_data1, fitted_lambda1 = stats.boxcox(project.Actual_Shipment_Time)

/usr/local/lib/python3.10/dist-packages/scipy/stats/_morestats.py in boxcox(x,␣
↪lmbda, alpha, optimizer)
   1103
   1104     # If lmbda=None, find the lmbda that maximizes the log-likelihood␣
↪function.
-> 1105     lmax = boxcox_normmax(x, method='mle', optimizer=optimizer)
   1106     y = boxcox(x, lmax)
   1107
```

```
/usr/local/lib/python3.10/dist-packages/scipy/stats/_morestats.py in␣
↪boxcox_normmax(x, brack, method, optimizer)
   1274
   1275     optimfunc = methods[method]
-> 1276     res = optimfunc(x)
   1277     if res is None:
   1278         message = ("`optimizer` must return an object containing the␣
↪optimal "

/usr/local/lib/python3.10/dist-packages/scipy/stats/_morestats.py in _mle(x)
   1259             return -boxcox_llf(lmb, data)
   1260
-> 1261         return _optimizer(_eval_mle, args=(x,))
   1262
   1263     def _all(x):

/usr/local/lib/python3.10/dist-packages/scipy/stats/_morestats.py in␣
↪_optimizer(func, args)
   1221
   1222         def _optimizer(func, args):
-> 1223             return optimize.brent(func, args=args, brack=brack)
   1224
   1225     # Otherwise check optimizer.

/usr/local/lib/python3.10/dist-packages/scipy/optimize/_optimize.py in␣
↪brent(func, args, brack, tol, full_output, maxiter)
   2640     options = {'xtol': tol,
   2641                'maxiter': maxiter}
-> 2642     res = _minimize_scalar_brent(func, brack, args, **options)
   2643     if full_output:
   2644         return res['x'], res['fun'], res['nit'], res['nfev']

/usr/local/lib/python3.10/dist-packages/scipy/optimize/_optimize.py in␣
↪_minimize_scalar_brent(func, brack, args, xtol, maxiter, disp,␣
↪**unknown_options)
   2677                    full_output=True, maxiter=maxiter, disp=disp)
   2678     brent.set_bracket(brack)
-> 2679     brent.optimize()
   2680     x, fval, nit, nfev = brent.get_result(full_output=True)
   2681

/usr/local/lib/python3.10/dist-packages/scipy/optimize/_optimize.py in␣
↪optimize(self)
   2447         # set up for optimization
   2448         func = self.func
-> 2449         xa, xb, xc, fa, fb, fc, funcalls = self.get_bracket_info()
   2450         _mintol = self._mintol
   2451         _cg = self._cg
```

```
/usr/local/lib/python3.10/dist-packages/scipy/optimize/_optimize.py in␣
↪get_bracket_info(self)
   2416             xa, xb, xc, fa, fb, fc, funcalls = bracket(func, args=args)
   2417         elif len(brack) == 2:
-> 2418             xa, xb, xc, fa, fb, fc, funcalls = bracket(func, xa=brack[0],
   2419                                                        xb=brack[1],␣
↪args=args)
   2420         elif len(brack) == 3:

/usr/local/lib/python3.10/dist-packages/scipy/optimize/_optimize.py in␣
↪bracket(func, xa, xb, args, grow_limit, maxiter)
   3046         e = BracketError(msg)
   3047         e.data = (xa, xb, xc, fa, fb, fc, funcalls)
-> 3048         raise e
   3049
   3050     return xa, xb, xc, fa, fb, fc, funcalls

BracketError: The algorithm terminated without finding a valid bracket. Consider␣
↪trying different initial points.
```

[68]:
```
fitted_data2, fitted_lambda2 = stats.boxcox(project.Planned_Shipment_Time)
```

[69]:
```
fitted_data3, fitted_lambda3 = stats.boxcox(project.Planned_Delivery_Time)
```

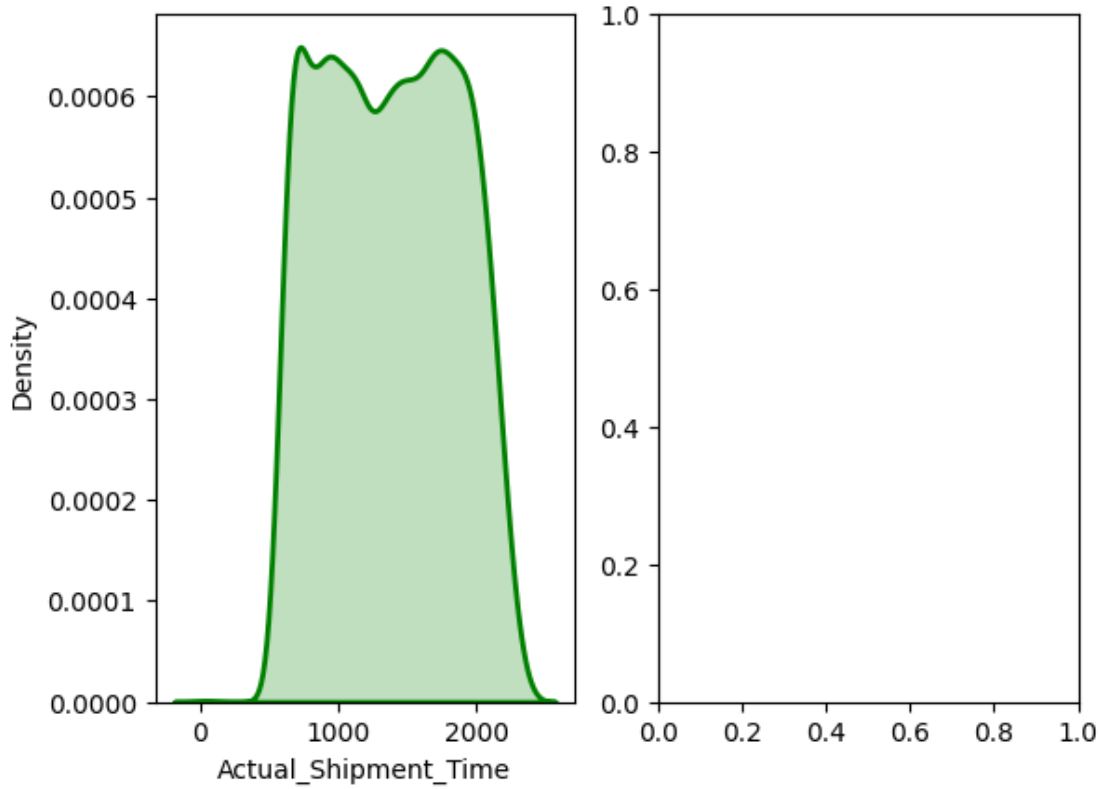
[70]:
```
fitted_data4, fitted_lambda4 = stats.boxcox(project.Planned_TimeofTravel)
```

[73]:
```
# creating axes to draw plots
fig, ax = plt.subplots(1 ,2)

# Plotting the original data (non-normal) and fitted data (normal)
sns.distplot(project.Actual_Shipment_Time, hist = False, kde = True,
             kde_kws = {'shade': True, 'linewidth': 2},
             label = "Non-Normal", color = "green", ax = ax[0])

sns.distplot(fitted_data1, hist = False, kde = True,
             kde_kws = {'shade': True, 'linewidth': 2},
             label = "Normal", color = "green", ax = ax[1])
# adding legends to the subplots
plt.legend(loc = "upper right")

# rescaling the subplots
fig.set_figheight(5)
fig.set_figwidth(10)
```

```
<ipython-input-73-4db82e9e50f1>:5: UserWarning:
```

```
`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density
plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(project.Actual_Shipment_Time, hist = False, kde = True,
/usr/local/lib/python3.10/dist-packages/seaborn/distributions.py:2511:
FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  kdeplot(**{axis: a}, ax=ax, color=kde_color, **kde_kws)
```

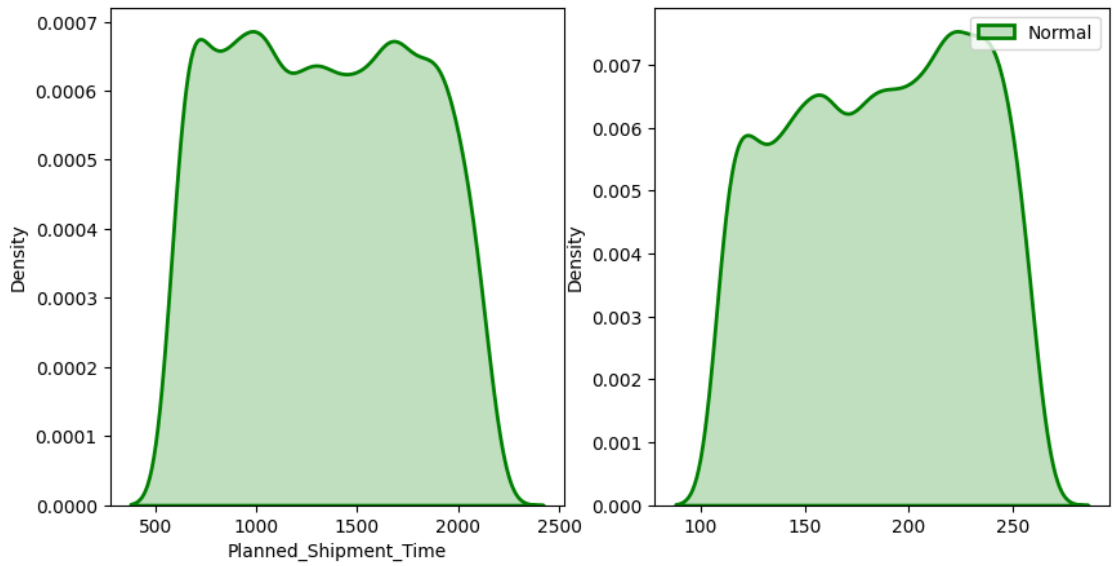
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-73-4db82e9e50f1> in <cell line: 9>()
      7                 label = "Non-Normal", color = "green", ax = ax[0])
      8
----> 9 sns.distplot(fitted_data1, hist = False, kde = True,
     10                 kde_kws = {'shade': True, 'linewidth': 2},
     11                 label = "Normal", color = "green", ax = ax[1])

NameError: name 'fitted_data1' is not defined
```



[75]:
```
# creating axes to draw plots
fig, ax = plt.subplots(1 ,2)
# Plotting the original data (non-normal) and fitted data (normal)
sns.distplot(project.Planned_Shipment_Time, hist = False, kde = True,
             kde_kws = {'shade': True, 'linewidth': 2},
             label = "Non-Normal", color = "green", ax = ax[0])

sns.distplot(fitted_data2, hist = False, kde = True,
             kde_kws = {'shade': True, 'linewidth': 2},
             label = "Normal", color = "green", ax = ax[1])
# adding legends to the subplots
plt.legend(loc = "upper right")

# rescaling the subplots
fig.set_figheight(5)
fig.set_figwidth(10)
```

```
<ipython-input-75-21443b738d73>:4: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
```

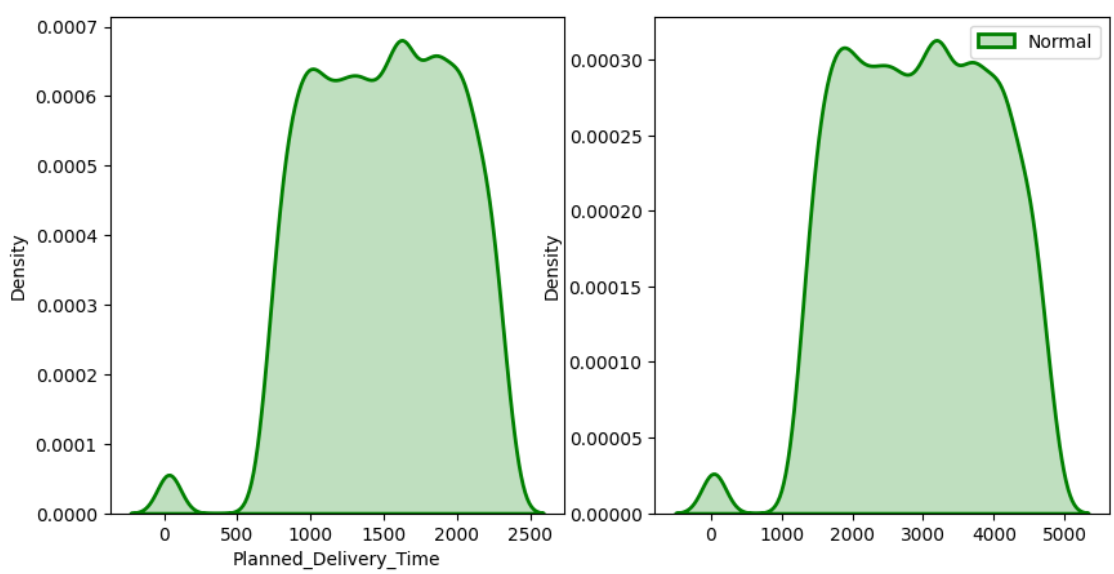
```
similar flexibility) or `kdeplot` (an axes-level function for kernel density
plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(project.Planned_Shipment_Time, hist = False, kde = True,
/usr/local/lib/python3.10/dist-packages/seaborn/distributions.py:2511:
FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  kdeplot(**{axis: a}, ax=ax, color=kde_color, **kde_kws)
<ipython-input-75-21443b738d73>:8: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density
plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(fitted_data2, hist = False, kde = True,
/usr/local/lib/python3.10/dist-packages/seaborn/distributions.py:2511:
FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  kdeplot(**{axis: a}, ax=ax, color=kde_color, **kde_kws)
```



[76]:
```
# creating axes to draw plots
fig, ax = plt.subplots(1 ,2)

# Plotting the original data (non-normal) and fitted data (normal)
sns.distplot(project.Planned_Delivery_Time, hist = False, kde = True,
             kde_kws = {'shade': True, 'linewidth': 2},
             label = "Non-Normal", color = "green", ax = ax[0])

sns.distplot(fitted_data3, hist = False, kde = True,
             kde_kws = {'shade': True, 'linewidth': 2},
             label = "Normal", color = "green", ax = ax[1])

# adding legends to the subplots
plt.legend(loc = "upper right")

# rescaling the subplots
fig.set_figheight(5)
fig.set_figwidth(10)
```

```
<ipython-input-76-d623d876a3c6>:5: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density
plots).

For a guide to updating your code to use the new functions, please see
```

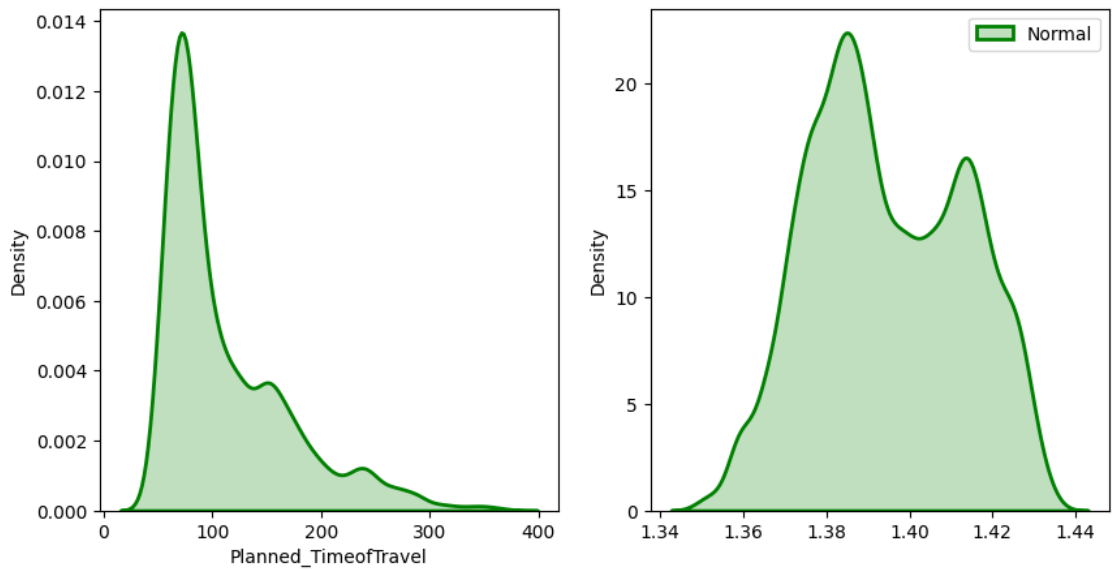
```
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(project.Planned_Delivery_Time, hist = False, kde = True,
/usr/local/lib/python3.10/dist-packages/seaborn/distributions.py:2511:
FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  kdeplot(**{axis: a}, ax=ax, color=kde_color, **kde_kws)
<ipython-input-76-d623d876a3c6>:9: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density
plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(fitted_data3, hist = False, kde = True,
/usr/local/lib/python3.10/dist-packages/seaborn/distributions.py:2511:
FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  kdeplot(**{axis: a}, ax=ax, color=kde_color, **kde_kws)
```



[77]:
```
# creating axes to draw plots
fig, ax = plt.subplots(1 ,2)

# Plotting the original data (non-normal) and fitted data (normal)
sns.distplot(project.Planned_TimeofTravel, hist = False, kde = True,
             kde_kws = {'shade': True, 'linewidth': 2},
             label = "Non-Normal", color = "green", ax = ax[0])

sns.distplot(fitted_data4, hist = False, kde = True,
             kde_kws = {'shade': True, 'linewidth': 2},
             label = "Normal", color = "green", ax = ax[1])

# adding legends to the subplots
plt.legend(loc = "upper right")

# rescaling the subplots
fig.set_figheight(5)
fig.set_figwidth(10)
```

```
<ipython-input-77-12dfa2effae1>:5: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density
plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(project.Planned_TimeofTravel, hist = False, kde = True,
/usr/local/lib/python3.10/dist-packages/seaborn/distributions.py:2511:
FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  kdeplot(**{axis: a}, ax=ax, color=kde_color, **kde_kws)
<ipython-input-77-12dfa2effae1>:9: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density
plots).
```

```
For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(fitted_data4, hist = False, kde = True,
/usr/local/lib/python3.10/dist-packages/seaborn/distributions.py:2511:
FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  kdeplot(**{axis: a}, ax=ax, color=kde_color, **kde_kws)
```



29 Transformed data

[79]:
```
print(f"Lambda value used for Transformation: {fitted_lambda1}")
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-79-e91fe5b15a3f> in <cell line: 1>()
----> 1 print(f"Lambda value used for Transformation: {fitted_lambda1}")

NameError: name 'fitted_lambda1' is not defined
```

[80]:
```
print(f"Lambda value used for Transformation: {fitted_lambda2}")
```

```
Lambda value used for Transformation: 0.6744165965976693
```

[81]:
```
print(f"Lambda value used for Transformation: {fitted_lambda3}")
```

```
Lambda value used for Transformation: 1.1053861600158499
```

[82]:
```
print(f"Lambda value used for Transformation: {fitted_lambda4}")
```

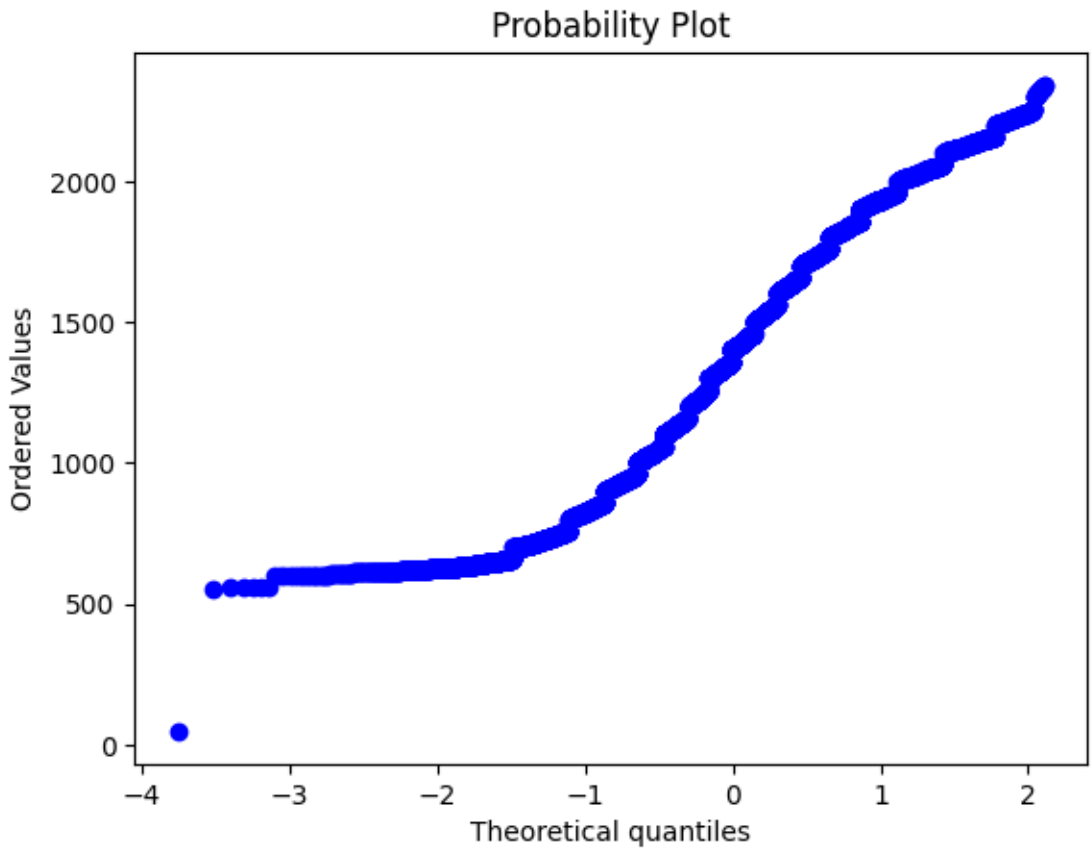
```
Lambda value used for Transformation: -0.685297253290949
```

30 Yeo-Johnson Transform

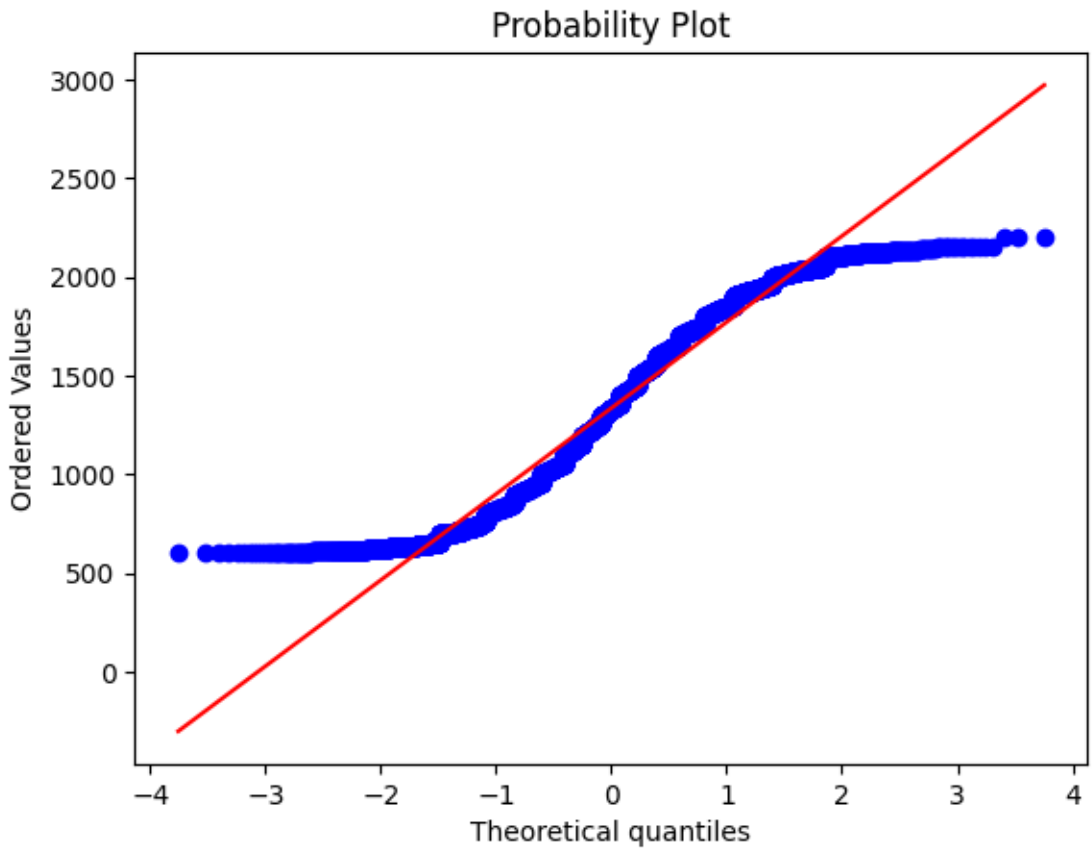
[83]:
```
# import modules
import pandas as pd
from scipy import stats

# Plotting modules
import seaborn as sns
import matplotlib.pyplot as plt
import pylab
```

[84]:
```
# Read data into Python
project = pd.read_csv(r"/content/Datasets.csv")
```

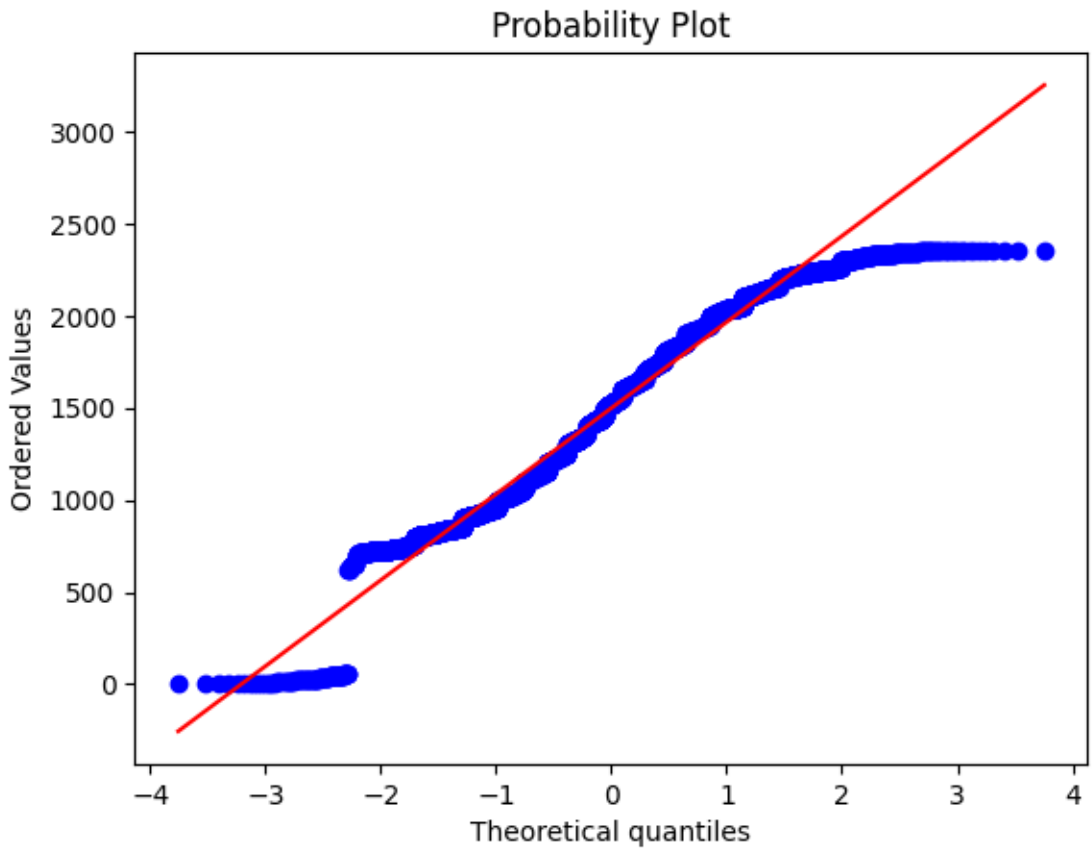
[85]:
```
# Original data

prob1 = stats.probplot(project.Actual_Shipment_Time, dist = stats.norm, plot =␣
↪pylab)
```



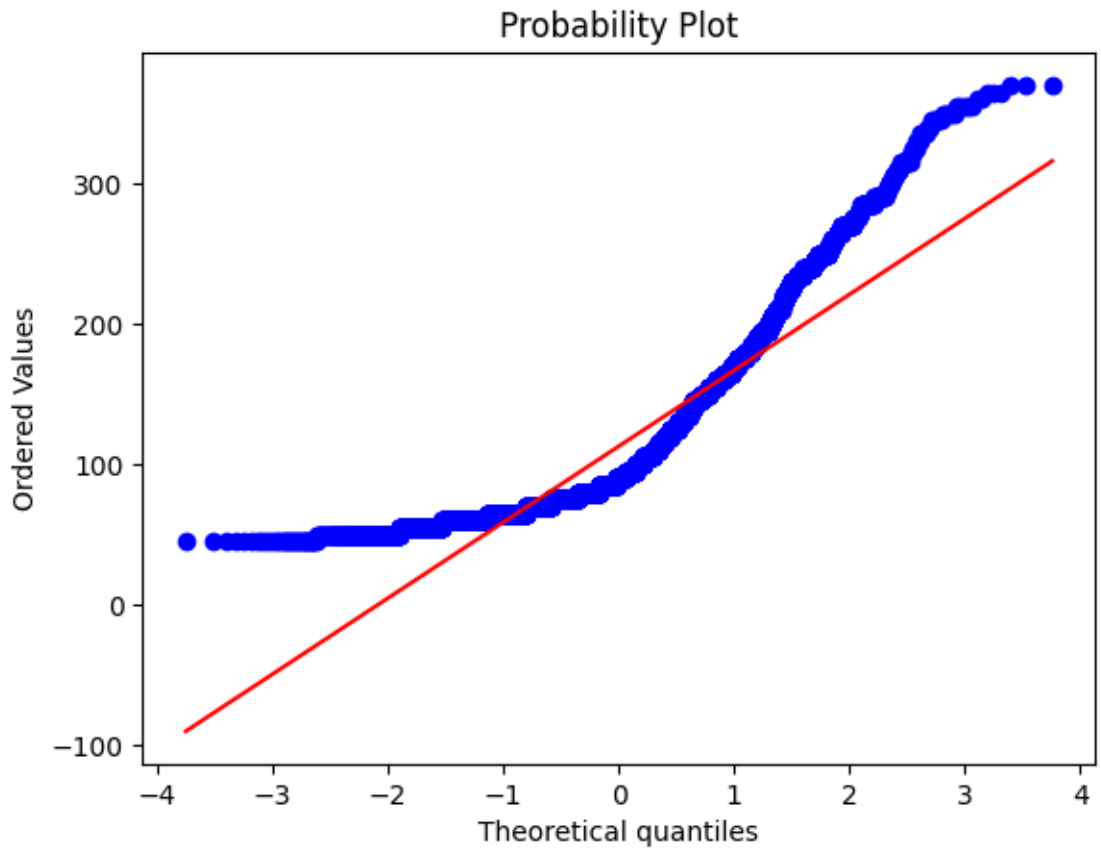
[86]:
```
prob2 = stats.probplot(project.Planned_Shipment_Time, dist = stats.norm, plot =␣
↪pylab)
```



[87]:
```
prob3 = stats.probplot(project.Planned_Delivery_Time, dist = stats.norm, plot =␣
↪pylab)
```



[88]:
```
prob4= stats.probplot(project.Planned_TimeofTravel, dist = stats.norm, plot =␣
↪pylab)
```



[89]:
```
pip install feature_engine
```

```
Collecting feature_engine
  Downloading feature_engine-1.6.2-py2.py3-none-any.whl (328 kB)
                              328.9/328.9

kB 6.9 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.18.2 in
/usr/local/lib/python3.10/dist-packages (from feature_engine) (1.23.5)
Requirement already satisfied: pandas>=1.0.3 in /usr/local/lib/python3.10/dist-
packages (from feature_engine) (1.5.3)
Requirement already satisfied: scikit-learn>=1.0.0 in
/usr/local/lib/python3.10/dist-packages (from feature_engine) (1.2.2)
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.10/dist-
packages (from feature_engine) (1.11.3)
Requirement already satisfied: statsmodels>=0.11.1 in
/usr/local/lib/python3.10/dist-packages (from feature_engine) (0.14.0)
Requirement already satisfied: python-dateutil>=2.8.1 in
/usr/local/lib/python3.10/dist-packages (from pandas>=1.0.3->feature_engine)
(2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
```

```
packages (from pandas>=1.0.3->feature_engine) (2023.3.post1)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn>=1.0.0->feature_engine) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-
learn>=1.0.0->feature_engine) (3.2.0)
Requirement already satisfied: patsy>=0.5.2 in /usr/local/lib/python3.10/dist-
packages (from statsmodels>=0.11.1->feature_engine) (0.5.3)
Requirement already satisfied: packaging>=21.3 in
/usr/local/lib/python3.10/dist-packages (from
statsmodels>=0.11.1->feature_engine) (23.2)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages
(from patsy>=0.5.2->statsmodels>=0.11.1->feature_engine) (1.16.0)
Installing collected packages: feature_engine
Successfully installed feature_engine-1.6.2
```

[90]:
```python
from feature_engine import transformation
```

[91]:
```python
# Set up the variable transformer

ts1 = transformation.YeoJohnsonTransformer(variables = 'Actual_Shipment_Time')

ts2 = transformation.YeoJohnsonTransformer(variables = 'Planned_Shipment_Time')

ts3 = transformation.YeoJohnsonTransformer(variables = 'Planned_Delivery_Time')

ts4 = transformation.YeoJohnsonTransformer(variables = 'Planned_TimeofTravel')
```

[94]:
```python
rx1 = ts1.fit_transform(project)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-94-158263fb888e> in <cell line: 1>()
----> 1 rx1 = ts1.fit_transform(project)

/usr/local/lib/python3.10/dist-packages/sklearn/utils/_set_output.py in␣
↪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

/usr/local/lib/python3.10/dist-packages/sklearn/base.py in fit_transform(self,␣
↪X, y, **fit_params)
    876         if y is None:
    877             # fit method of arity 1 (unsupervised transformation)
```

```
--> 878             return self.fit(X, **fit_params).transform(X)
    879         else:
    880             # fit method of arity 2 (supervised transformation)

/usr/local/lib/python3.10/dist-packages/feature_engine/transformation/yeojohnso
 ↪py in fit(self, X, y)
    129
    130         # check input dataframe
--> 131         X = super().fit(X)
    132
    133         self.lambda_dict_ = {}

/usr/local/lib/python3.10/dist-packages/feature_engine/_base_transformers/
 ↪base_numerical.py in fit(self, X)
     62
     63         # check if dataset contains na or inf
---> 64         _check_contains_na(X, self.variables_)
     65         _check_contains_inf(X, self.variables_)
     66

/usr/local/lib/python3.10/dist-packages/feature_engine/dataframe_checks.py in␣
 ↪_check_contains_na(X, variables)
    266
    267     if X[variables].isnull().any().any():
--> 268         raise ValueError(
    269             "Some of the variables in the dataset contain NaN. Check an␣
 ↪"
    270             "remove those before using this transformer."

ValueError: Some of the variables in the dataset contain NaN. Check and remove␣
 ↪those before using this transformer.
```

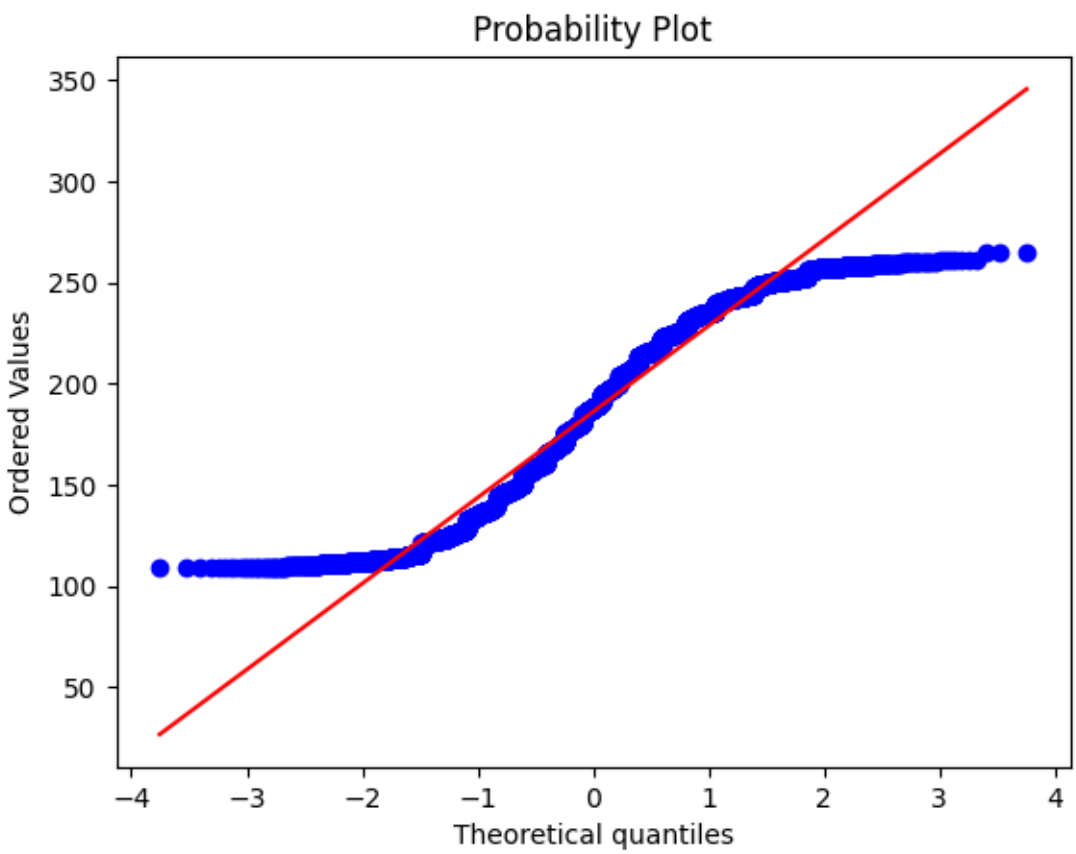
[93]:
```
rx2 = ts2.fit_transform(project)

rx3 = ts3.fit_transform(project)

rx4 = ts4.fit_transform(project)
```

[95]:
```
# Transformed data

prob1 = stats.probplot(rx1.Actual_Shipment_Time, dist = stats.norm, plot =␣
 ↪pylab)

prob2 = stats.probplot(rx2.Planned_Shipment_Time, dist = stats.norm, plot =␣
 ↪pylab)
```

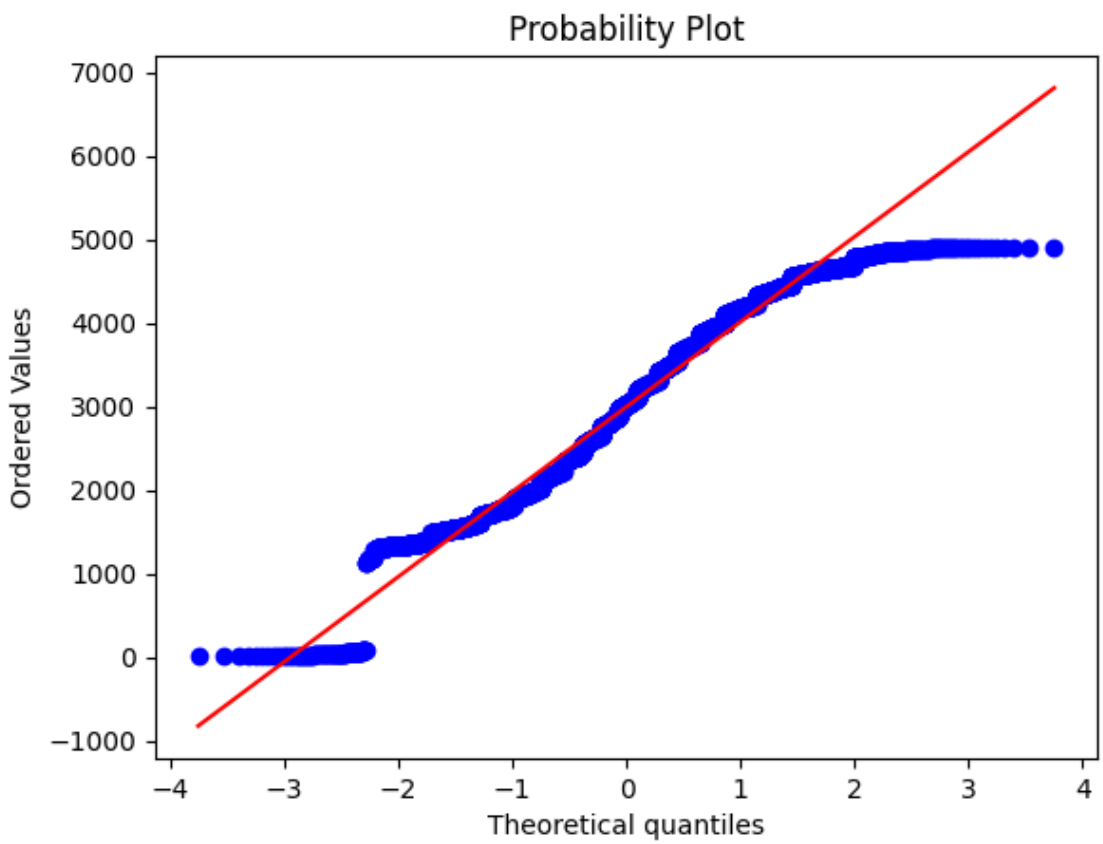
```
prob3 = stats.probplot(rx3.Planned_Delivery_Time, dist = stats.norm, plot =␣
↪pylab)

prob4 = stats.probplot(rx4.Planned_TimeofTravel, dist = stats.norm, plot =␣
↪pylab)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-95-9cd511449d88> in <cell line: 3>()
      1 # Transformed data
      2
----> 3 prob1 = stats.probplot(rx1.Actual_Shipment_Time, dist = stats.norm, plot␣
↪= pylab)
      4
      5 prob2 = stats.probplot(rx2.Planned_Shipment_Time, dist = stats.norm,␣
↪plot = pylab)

NameError: name 'rx1' is not defined
```

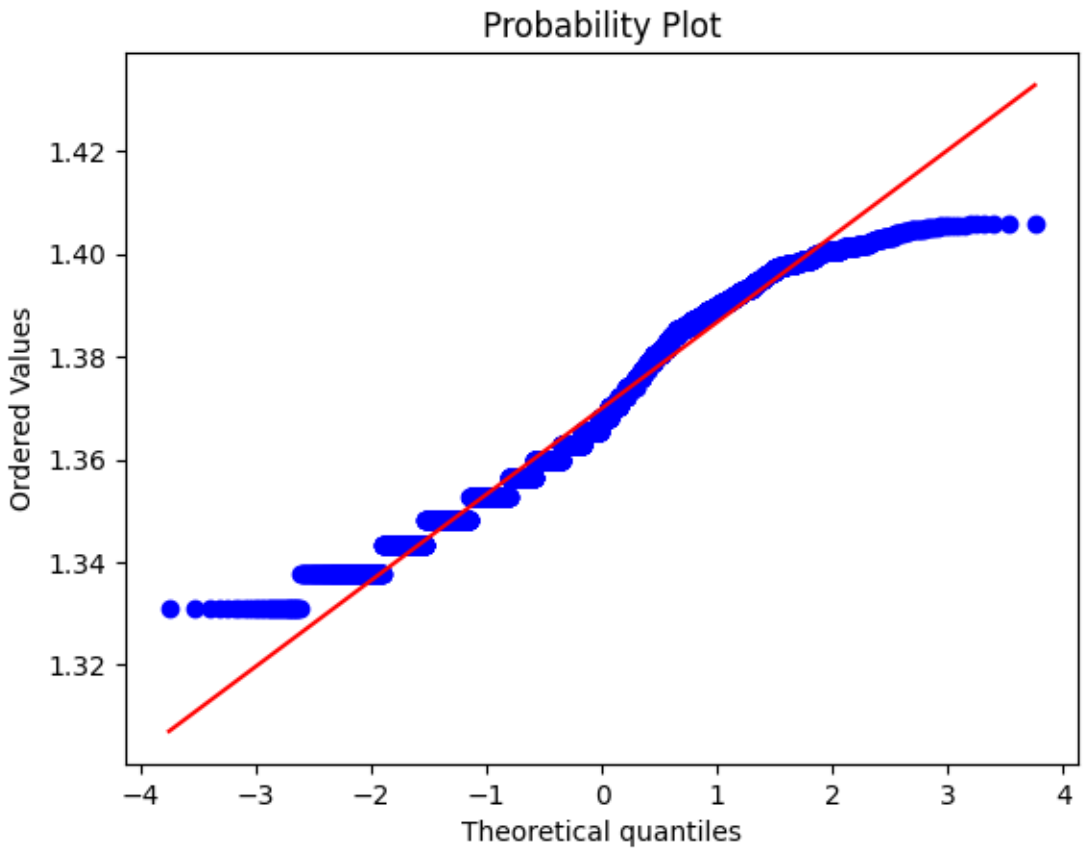
[96]:
```
prob2 = stats.probplot(rx2.Planned_Shipment_Time, dist = stats.norm, plot =␣
↪pylab)
```



[97]:
```
prob3 = stats.probplot(rx3.Planned_Delivery_Time, dist = stats.norm, plot =␣
↪pylab)
```



[98]:
```
prob4 = stats.probplot(rx4.Planned_TimeofTravel, dist = stats.norm, plot =␣
↪pylab)
```



Standardization and Normalization

[99]:
```
import pandas as pd
import numpy as np
```

[100]:
```
project = pd.read_csv(r"/content/Datasets.csv")
```

[101]:
```
ps = project.describe()
```

[102]:
```
### Standardization
from sklearn.preprocessing import StandardScaler
```

[104]:
```
# Initialise the Scaler
scaler = StandardScaler()
```

[105]:
```
# To scale data
dn = scaler.fit_transform(project)
# Convert the array back to a dataframe
dataset = pd.DataFrame(dn)
res = dataset.describe()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-105-d7d61a45e15e> in <cell line: 2>()
      1 # To scale data
----> 2 dn = scaler.fit_transform(project)
      3 # Convert the array back to a dataframe
      4 dataset = pd.DataFrame(dn)
      5 res = dataset.describe()

/usr/local/lib/python3.10/dist-packages/sklearn/utils/_set_output.py in␣
↪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

/usr/local/lib/python3.10/dist-packages/sklearn/base.py in fit_transform(self,␣
↪X, y, **fit_params)
    876         if y is None:
    877             # fit method of arity 1 (unsupervised transformation)
--> 878             return self.fit(X, **fit_params).transform(X)
    879         else:
    880             # fit method of arity 2 (supervised transformation)

/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_data.py in␣
↪fit(self, X, y, sample_weight)
    822         # Reset internal state before fitting
    823         self._reset()
--> 824         return self.partial_fit(X, y, sample_weight)
    825
    826     def partial_fit(self, X, y=None, sample_weight=None):

/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_data.py in␣
↪partial_fit(self, X, y, sample_weight)
    859
    860         first_call = not hasattr(self, "n_samples_seen_")
--> 861         X = self._validate_data(
    862             X,
    863             accept_sparse=("csr", "csc"),

/usr/local/lib/python3.10/dist-packages/sklearn/base.py in _validate_data(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:

/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py in␣
↪check_array(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)
    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(

/usr/local/lib/python3.10/dist-packages/sklearn/utils/_array_api.py in␣
↪_asarray_with_order(array, dtype, order, copy, xp)
    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:

/usr/local/lib/python3.10/dist-packages/pandas/core/generic.py in␣
↪__array__(self, dtype)
   2068
   2069     def __array__(self, dtype: npt.DTypeLike | None = None) -> np.
↪ndarray:
-> 2070         return np.asarray(self._values, dtype=dtype)
   2071
   2072     def __array_wrap__(

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

[106]:
```
# Normalization
''' Alternatively we can use the below function'''
from sklearn.preprocessing import MinMaxScaler
minmaxscale = MinMaxScaler()

dn_n = minmaxscale.fit_transform(dn)
dataset1 = pd.DataFrame(dn_n)

res1 = dataset1.describe()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-106-df2a998530e8> in <cell line: 6>()
```

```
      4 minmaxscale = MinMaxScaler()
      5
----> 6 dn_n = minmaxscale.fit_transform(dn)
      7 dataset1 = pd.DataFrame(dn_n)
      8

NameError: name 'dn' is not defined
```

[108]:
```
### Normalization
## load dataset
project = pd.read_csv(r"/content/Datasets.csv")

project.columns
project.drop([ 'Actual_Shipment_Time','Planned_Shipment_Time',␣
 ↪'Planned_Delivery_Time', 'Carrier_Name',
       'Carrier_Num', 'Planned_TimeofTravel', ], axis = 1, inplace = True)
```

[109]:
```
a2 = project.describe()
```

[110]:
```
# Get dummies
ethnic1 = pd.get_dummies(project, drop_first = True)

a3 = ethnic1.describe()
```

[111]:
```
### Normalization function - Custom Function
# Range converts to: 0 to 1
def norm_func(i):
    x = (i-i.min())/(i.max()-i.min())
    return(x)

df_norm = norm_func(ethnic1)
b = df_norm.describe()
```

[112]:
```
''' Alternatively we can use the below function'''
from sklearn.preprocessing import MinMaxScaler
minmaxscale = MinMaxScaler()

ethnic1_minmax = minmaxscale.fit_transform(ethnic1)
df_ethnic1 = pd.DataFrame(ethnic1_minmax)
minmax_res = df_ethnic1.describe()
```

[113]:
```
'''Robust Scaling
Scale features using statistics that are robust to outliers'''

from sklearn.preprocessing import RobustScaler
```

```
robust_model = RobustScaler()

df_robust = robust_model.fit_transform(ethnic1)

dataset_robust = pd.DataFrame(df_robust)
res_robust = dataset_robust.describe()
```

[114]:
```
import pandas as pd
```

31 clean data

[115]:
```
project = pd.read_csv(r"/content/Datasets.csv")
```

[116]:
```
print(f"Cleaned data saved to: {project}")
```

```
Cleaned data saved to:       Year  Month  DayofMonth  DayOfWeek
Actual_Shipment_Time  \
0     2008      1           3          4                2003.0
1     2008      1           3          4                 754.0
2     2008      1           3          4                 628.0
3     2008      1           3          4                 926.0
4     2008      1           3          4                1829.0
…     …     …           …          …                     …
7994  2008      1           5          6                1534.0
7995  2008      1           5          6                1200.0
7996  2008      1           5          6                 902.0
7997  2008      1           5          6                1722.0
7998  2008      1           5          6                 721.0

      Planned_Shipment_Time  Planned_Delivery_Time Carrier_Name  Carrier_Num  \
0                      1955                   2225           WN          335
1                       735                   1000           WN         3231
2                       620                    750           WN          448
3                       930                   1100           WN         1746
4                      1755                   1925           WN         3920
…                       …                      …           …           …
7994                   1520                   1620           WN         1516
7995                   1200                   1255           WN         2621
7996                    900                   1000           WN         3569
7997                   1715                   1930           WN          383
7998                    715                    930           WN         1945

      Planned_TimeofTravel  Shipment_Delay Source Destination  Distance  \
0                      150             8.0    IAD         TPA       810
1                      145            19.0    IAD         TPA       810
2                       90             8.0    IND         BWI       515
```

```
3                       90            -4.0    IND         BWI       515
4                       90            34.0    IND         BWI       515
…                     …             …    …         …       …
7994                    60            14.0    RDU         BWI       255
7995                    55             0.0    RDU         BWI       255
7996                    60             2.0    RDU         BWI       255
7997                   315             7.0    RDU         LAS      2027
7998                   315             6.0    RDU         LAS      2027

      Delivery_Status
0                 0.0
1                 1.0
2                 0.0
3                 0.0
4                 1.0
…               …
7994              0.0
7995              0.0
7996              0.0
7997              0.0
7998              0.0

[7999 rows x 15 columns]
```

[ ]:

```
```