Imports

```
# Imports
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt


df = pd.read_csv('Pred_Ast_Diam_2.csv')
```

The Dataset

[ ] ↳ *8 cells hidden*

Model Implementation

```
import pandas as pd
import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"] = (20,10)
from sklearn import preprocessing
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier



df = pd.read_csv('Pred_Ast_Diam_2.csv')
```

```
df.head(25)
```

	orbit_id	e	a	i	om	w
0	JPL 35	0.242027	2.201791	2.536221	313.311389	18.989048
1	JPL 25	0.256856	2.338209	22.326589	10.489602	105.115594
2	JPL 28	0.160543	2.228812	1.747387	121.579382	252.465454
3	JPL 35	0.167945	2.241299	2.428619	161.636895	172.846491
4	JPL 34	0.253295	2.467536	6.757106	137.130656	259.158793

```
neworder = ['orbit_id', 'e', 'a', 'i', 'om', 'w', 'ma', 'n', 'tp', 'moid', 'moid_jup', 'class', 'producer', 'data_arc', 'n_obs_used',
df=df.reindex(columns=neworder)
data_to_use = df
del data_to_use['orbit_id']
del data_to_use['class']
del data_to_use['producer']
data_to_use.dropna(inplace=True) # find null values, delete row
data_to_use.isnull().sum()
```

```
e                 0
a                 0
i                 0
om                0
w                 0
ma                0
n                 0
tp                0
moid              0
moid_jup          0
data_arc          0
n_obs_used        0
rms               0
albedo            0
diameter_sigma    0
first_year_obs    0
first_month_obs   0
last_obs_year     0
last_obs_month    0
```

```
diameter           0
dtype: int64
```

```
df.corr()['diameter'].abs().sort_values(ascending=False)
```

```
diameter           1.000000
moid               0.472688
n                  0.434750
moid_jup           0.411625
a                  0.403511
n_obs_used         0.360070
data_arc           0.301341
first_year_obs     0.281194
albedo             0.264930
last_obs_year      0.233702
tp                 0.141849
e                  0.128467
rms                0.118123
i                  0.105850
last_obs_month     0.067802
first_month_obs    0.040719
ma                 0.013657
diameter_sigma     0.008756
w                  0.003206
om                 0.002646
Name: diameter, dtype: float64
```

```
from sklearn.model_selection import train_test_split
predictors = df.drop('diameter',axis=1)
target = df['diameter']
X_train,X_test,Y_train,Y_test = train_test_split(predictors,target,test_size=0.20,random_state=0)
```

```
X_train.head()
```

	e	a	i	om	w	
107438	0.216103	2.946804	4.957513	16.571451	201.688379	324.3
11941	0.055395	2.681718	5.399944	66.345594	150.260650	270.8
92062	0.110799	2.659429	11.284872	26.228985	186.273281	161.1
40336	0.252554	2.407456	8.457145	274.538568	21.484548	155.1

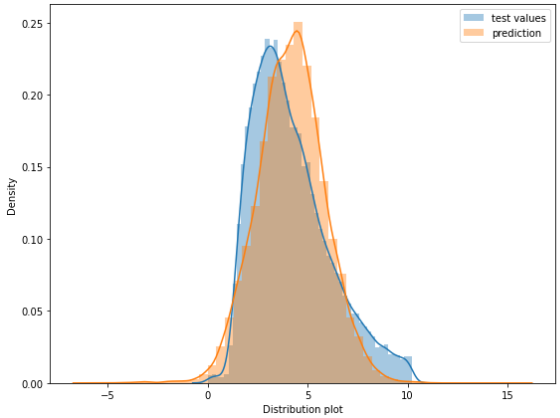
```
from sklearn import preprocessing

#Input standard normalization:
std_scaler = preprocessing.StandardScaler().fit(X_train)

def scaler(X):
    x_norm_arr= std_scaler.fit_transform(X)
    return pd.DataFrame(x_norm_arr, columns=X.columns, index = X.index)

X_train_norm = scaler(X_train)
X_test_norm = scaler(X_test)

def inverse_scaler(X):
    x_norm_arr= std_scaler.inverse_transform(X)
    return pd.DataFrame(x_norm_arr, columns=X.columns, index = X.index)


from sklearn.metrics import r2_score
import seaborn as sns

def plot(prediction):
    fig, (ax1, ax2) = plt.subplots(1, 2,figsize=(20,7))
    sns.distplot(Y_test.values,label='test values', ax=ax1)
    sns.distplot(prediction ,label='prediction', ax=ax1)
    ax1.set_xlabel('Distribution plot')
    ax2.scatter(Y_test,prediction, c='orange',label='predictions')
    ax2.plot(Y_test,Y_test,c='blue',label='y=x')
    ax2.set_xlabel('test value')
    ax2.set_ylabel('estimated $\log(radius)$')
```

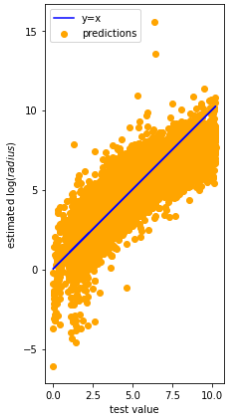
```
        ax1.legend()
        ax2.legend()
        ax2.axis('scaled') #same x y scale
    def score(prediction):
        score = r2_score(prediction,Y_test)
        return score
    def announce(score):
        print('The R^2 score achieved using this regression is:', round(score,3))
    algorithms = []
    scores = []


    # Linear Regression
    from sklearn.linear_model import LinearRegression
    lr = LinearRegression()

    # Training
    lr.fit(X_train,Y_train)

    # Predicting
    Y_pred_lr = lr.predict(X_test)

    # Scoring
    score_lr = score(Y_pred_lr)
    announce(score_lr)

    algorithms.append('LR')
    scores.append(score_lr)
```

```
     The R^2 score achieved using this regression is: 0.676
```

```
    plot(Y_pred_lr)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2
  warnings.warn(msg, FutureWarning)
```





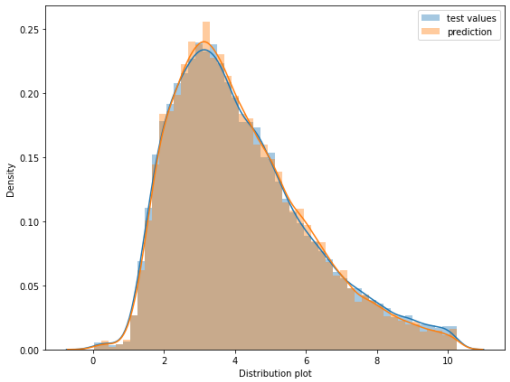

```
# Decision Tree
from sklearn import tree
decTree = tree.DecisionTreeRegressor()

# Training
decTree = decTree.fit(X_train_norm,Y_train)

# Predicting
Y_pred_tree = decTree.predict(X_test_norm)

# Scoring
score_tree = score(Y_pred_tree)
announce(score_tree)
```

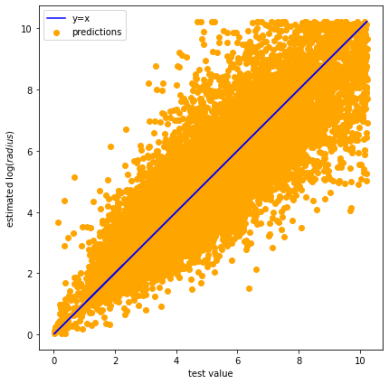
```
algorithms.append('DTree')
scores.append(score_tree)
```

```
The R^2 score achieved using this regression is: 0.771
```

```
plot(Y_pred_tree)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2
  warnings.warn(msg, FutureWarning)
```



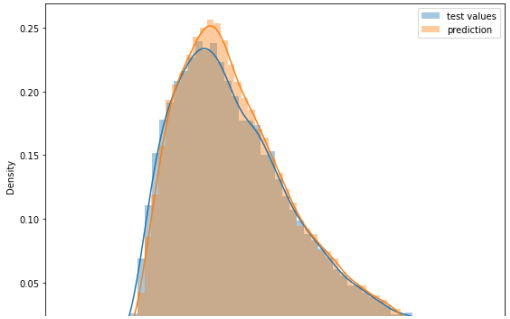


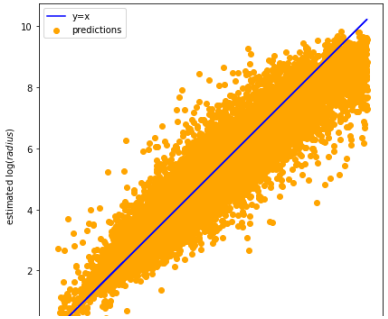
```
# Random Forest Regression
from sklearn.ensemble import RandomForestRegressor
forest = RandomForestRegressor(max_depth=32, n_estimators=50)

# Training
forest.fit(X_train_norm,np.ravel(Y_train))

# Predicting
Y_pred_forest = forest.predict(X_test_norm)

# Scoring
score_forest = score(Y_pred_forest)
announce(score_forest)

algorithms.append('RForest')
scores.append(score_forest)
```

```
     The R^2 score achieved using this regression is: 0.883
```

```
plot(Y_pred_forest)
```

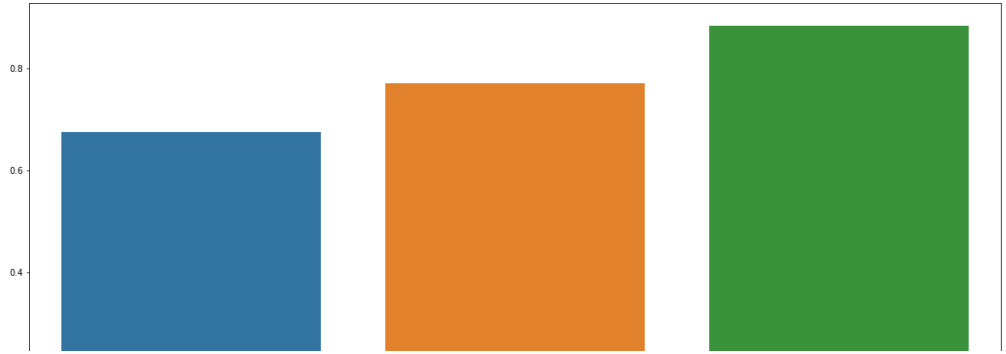
```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2
  warnings.warn(msg, FutureWarning)
```






```
sns.barplot(algorithms,scores)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fca0e856c50>
```



Double-click (or enter) to edit

Shapely Plots

```
#load in the data from local file

df = pd.read_csv('/content/Pred_Ast_Diam_2.csv')
print(df.shape)
df.head()
```

```
(126497, 23)
```

```
orbit_id  e  a  i  om  w
```

```
#Start with splitting the data into a train, validation, and test case using an 80/20 split.

from sklearn.model_selection import train_test_split
# Split into Train and Test sets
train, test = train_test_split(df, train_size=.80, test_size=0.20, random_state=42)

# Split train into train & val
train, val = train_test_split(train, train_size=0.80, test_size=0.20, random_state=42)

train.shape, val.shape, test.shape

### What would be the system for time dependent on first observed

#train = df[df.Date.dt.year <= 2016]
#val = df[df.Date.dt.year == 2017]
#test = df[df.Date.dt.year >= 2018]
```

```
((80957, 23), (20240, 23), (25300, 23))
```

```
!pip install shap
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: shap in /usr/local/lib/python3.7/dist-packages (0.41.0)
Requirement already satisfied: slicer==0.0.7 in /usr/local/lib/python3.7/dist-packages (from shap) (0.0.7)
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from shap) (1.3.5)
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.7/dist-packages (from shap) (1.5.0)
Requirement already satisfied: numba in /usr/local/lib/python3.7/dist-packages (from shap) (0.56.0)
Requirement already satisfied: packaging>20.9 in /usr/local/lib/python3.7/dist-packages (from shap) (21.3)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packages (from shap) (1.0.2)
Requirement already satisfied: tqdm>4.25.0 in /usr/local/lib/python3.7/dist-packages (from shap) (4.64.0)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from shap) (1.7.3)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from shap) (1.21.6)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.7/dist-packages (from packaging>20.9->shap) (
Requirement already satisfied: llvmlite<0.40,>=0.39.0dev0 in /usr/local/lib/python3.7/dist-packages (from numba->shap) (0.39.0)
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/dist-packages (from numba->shap) (4.12.0)
```

```
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-packages (from numba->shap) (57.4.0)
Requirement already satisfied: typing-extensions>=3.6.4 in /usr/local/lib/python3.7/dist-packages (from importlib-metadata->num
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from importlib-metadata->numba->shap) (3.8.
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (from pandas->shap) (2022.2.1)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas->shap) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.7.3->pandas->shap) (
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from scikit-learn->shap) (3.1.0)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from scikit-learn->shap) (1.1.0)
```

```
# Assign to X, y
target = 'diameter'
features = train.columns.drop('diameter')

X_train = train[features]
y_train = train[target]

X_val = val[features]
y_val = val[target]

X_test = test[features]
y_test = test[target]


import category_encoders as ce

ord_encoder = ce.OrdinalEncoder(cols = ['orbit_id'])
X_train_ordencoded = ord_encoder.fit_transform(X_train)
X_val_ordencoded = ord_encoder.transform(X_val)
X_test_ordencoded = ord_encoder.transform(X_test)

oh_encoder = ce.OneHotEncoder(use_cat_names=True, cols=['class','producer'])
X_train_encoded = oh_encoder.fit_transform(X_train_ordencoded)
X_val_encoded = oh_encoder.transform(X_val_ordencoded)
X_test_encoded = oh_encoder.transform(X_test_ordencoded)


# Get an individual observation to explain.
```

```
# For example, the 0th row from the test set.
row = X_test_encoded.iloc[[0]]
row
```

	orbit_id	e	a	i	om	
119557	9.0	0.066676	2.720769	18.546595	31.706207	44.8383

1 rows × 35 columns

```
# What was the actual diameter for this asteroid?
y_test.iloc[[0]]
```

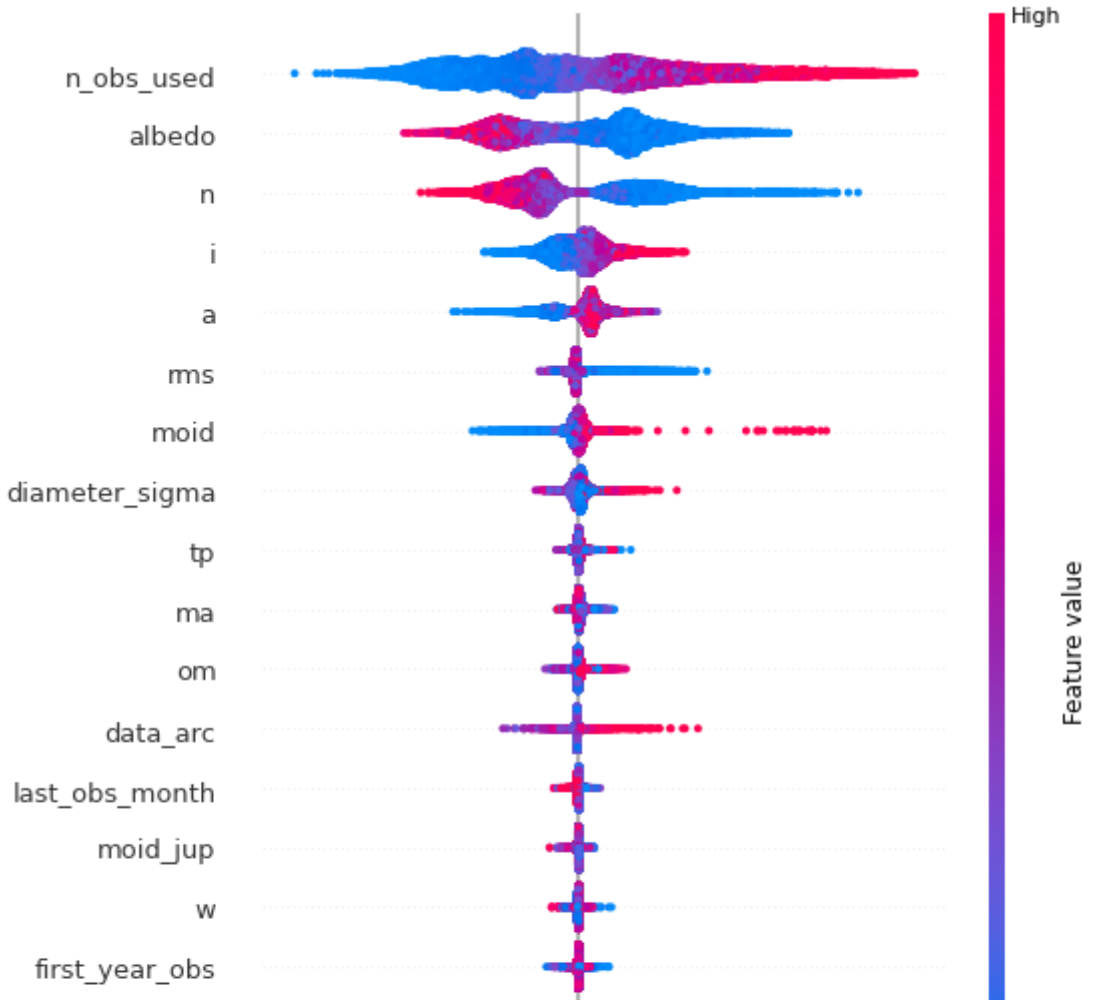
```
119557    2.242
Name: diameter, dtype: float64
```

```
import category_encoders as ce
from sklearn.model_selection import cross_val_score
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import RandomizedSearchCV

ord_encoder = ce.OrdinalEncoder(cols = ['orbit_id'])
X_train_ordencoded = ord_encoder.fit_transform(X_train)
X_val_ordencoded = ord_encoder.transform(X_val)
X_test_ordencoded = ord_encoder.transform(X_test)

oh_encoder = ce.OneHotEncoder(use_cat_names=True, cols=['class','producer'])
```

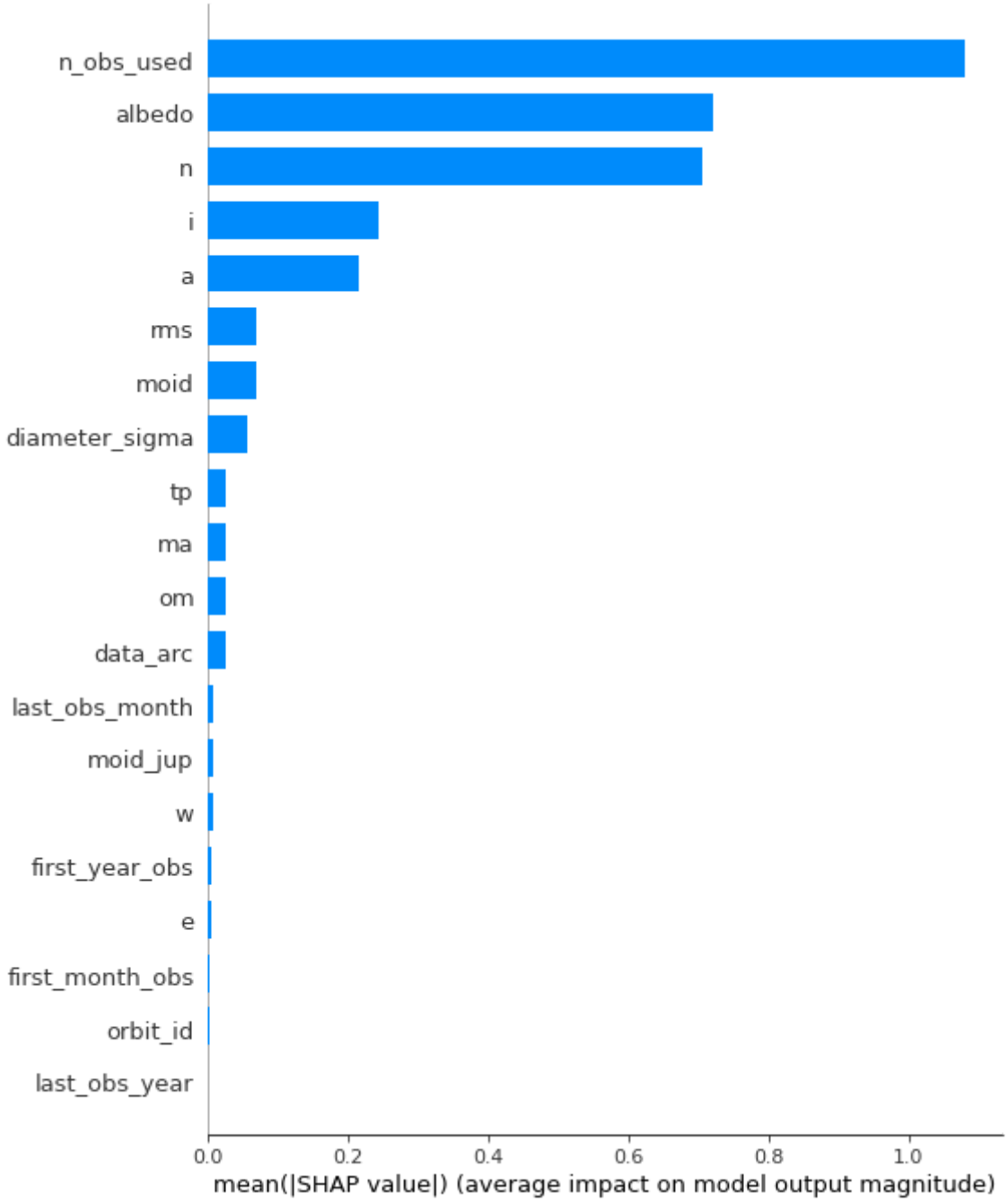
```
X_train_encoded = oh_encoder.fit_transform(X_train_ordencoded)
X_val_encoded = oh_encoder.transform(X_val_ordencoded)
X_test_encoded = oh_encoder.transform(X_test_ordencoded)

scaler = StandardScaler()
scaler.fit(X_train_encoded)
scaler.fit(X_val_encoded)
scaler.fit(X_test_encoded)

model_dt_shap = DecisionTreeRegressor(criterion='mse', max_depth=15, min_samples_leaf=15, min_samples_split=4, random_state=42)


model_dt_shap.fit(X_train_encoded,y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/tree/_classes.py:363: FutureWarning: Criterion 'mse' was deprecated in v1.0 and
  FutureWarning,
DecisionTreeRegressor(criterion='mse', max_depth=15, min_samples_leaf=15,
                      min_samples_split=4, random_state=42)
```

```
import shap
explainer = shap.TreeExplainer(model_dt_shap)
shap_values = explainer.shap_values(X_test_encoded)


# summarize the effects of all the features
shap.summary_plot(shap_values, X_test_encoded)
```




```
shap.summary_plot(shap_values, X_test_encoded, plot_type="bar")
```


n_obs_used
albedo
n
i
a
rms
moid
diameter_sigma
tp
ma
om
data_arc
last_obs_month
moid_jup
w
first_year_obs
e
first_month_obs
orbit_id
last_obs_year
0.0
0.2
0.4
0.6
0.8
1.0
mean(|SHAP value|) (average impact on model output magnitude)

✓ 0s completed at 10:25 PM