In [153]:
```
#Stars and stuff
from astroquery.gaia import Gaia
from astropy.io.votable import parse, parse_single_table
from astropy.timeseries import LombScargle
import pandas as pd

#Math
import numpy as np
from scipy.linalg import lstsq

#Machine Learning
from sklearn.model_selection import train_test_split

#Matplotlib
import matplotlib.pyplot as plt

#Caching
from joblib import Memory
location = "./cachedir"
memory = Memory(location, verbose=0)

#File manipulation
from urllib.request import urlopen
import io
from glob import glob
import os
```

In [2]:
```
def get_gaia_query_rrlyrae(num_stars = 100, num_clean_epochs = 40, conds
=None, verbose=False):
    add = ""
    if conds is not None:
        add = f"AND {conds}"
    query = f'''
        SELECT TOP {num_stars} *
        FROM gaiadr2.gaia_source as gaia
        JOIN gaiadr2.vari_rrlyrae using (source_id)
        WHERE
            num_clean_epochs_g > {num_clean_epochs}
    ''' + add
    if verbose:
        print(query)
    job = Gaia.launch_job_async(query)
    return job.get_results()

get_gaia_query_rrlyrae_cached = memory.cache(get_gaia_query_rrlyrae)
```

In [117]:
```
def get_gaia_query_general(query):
    job = Gaia.launch_job_async(query)
    return job.get_results()
get_gaia_query_general_cached = memory.cache(get_gaia_query_general)
```

In [3]:
```
def rmse(a, b):
    return np.sqrt(np.mean(np.square(a-b)))
```

In [4]:
```
def mse(a,b):
    return np.mean(np.square(a-b))
```

In [63]:
```
def get_Gband(data,index=None,source_id=None):
    assert index!=None or source_id!=None, "Must pass in either index or
source_id"
    if index!=None:
        if source_id!=None:
            print(f"Using index: {index} instead of source_id: {source_i
d}")
        selected_row=data[index]
        source_id=selected_row['source_id']
        print(f"Analyzing star with source_id: {source_id}")
    lc_f = f"lightcurve_xmls/{source_id}.xml"
    if os.path.exists(lc_f):
        votable = parse_single_table(lc_f)
    else:
        url = selected_row['epoch_photometry_url']
        if type(url) == bytes:
            url = url.decode('utf-8')
        votable = parse(url)
        votable.format = 'binary'
        with open(lc_f, 'w') as d:
            votable.to_xml(d)
        votable = parse_single_table(lc_f)
    Gstring = "G" if os.name == 'nt' else b'G'
    Gband = votable.array[votable.array['band'] == Gstring]
    return Gband
```

In [105]:
```
type(rrlyrae_rrc['pf'][0])==np.ma.core.MaskedConstant
```

Out[105]: True

In [106]:
```
def get_maxmin_freqs(data):
    if any([type(x)==np.ma.core.MaskedConstant for x in data['pf']]):
        col_name = 'p1_o'
    else:
        col_name = 'pf'
    max_freq, min_freq = 1/np.min(data[col_name]), 1/np.max(data[col_nam
e])
    return max_freq, min_freq
```

In [73]:
```
def estimate_period(Gband, max_freq=2.465, min_freq=1.044, p=False):
    mag = Gband['mag']
    flux = Gband['flux']
    time = Gband['time']
    flux_err = Gband['flux_error']

    freq, power = LombScargle(time, flux, flux_err).autopower(maximum_fr
equency=max_freq,
                                                              minimum_fre
quency=min_freq,
                                                              nyquist_fac
tor=100)
    period = 1/freq[np.argmax(power)]

    phase = time % period
    if p:
        plt.figure(figsize=(8,5))
        plt.plot(freq, power, '-k')
        plt.xlabel("Frequency")
        plt.ylabel("Spectral Power")

        fig, (ax1,ax2) = plt.subplots(1,2, figsize=(16,5))

        ax1.scatter(phase, flux)
        ax1.set(xlabel="Phase", ylabel="Flux")
        ax1.grid()

        ax2.scatter(phase, mag)
        ax2.set(xlabel="Magnitude", ylabel="Flux")
        ax2.grid()

        plt.show()

        print(f"Estimated period: 1/{freq[np.argmax(power)]:.5f} = {peri
od:.5f}")
#         print(f"Period as reported by vari_rrlyrae: {recorded_period:.
5f}")
#         print(f"RMSE: {np.sqrt(np.mean(np.square(period - recorded_per
iod)))}")

    return period
```

In [7]:
```
def plot_magnitude(Gband):
    time = Gband['time']
    mag = Gband['mag']
    mag_uncertainty = 1.09/Gband['flux_over_error']

    plt.figure(figsize=(8,5))
    plt.title("Magnitude w/ Magnitude Uncertainty")
    plt.fill_between(time, mag+mag_uncertainty/2,mag-mag_uncertainty/2,
color='blue', alpha=0.5)
    plt.xlabel("Time")
    plt.ylabel("Magnitude")
    plt.grid()
    plt.show()
    print(f"Estimated mean: {np.log(np.average(np.exp(mag)))}")
```

In [8]:
```
def setup_fit(flux, time, omega, k):
    num_samps = len(time)
    k_s = np.arange(1,k+1)
    tk_tiling = np.outer(time, k_s)*omega
    X = np.zeros((num_samps, 2*k+1))
    X[:,0] = np.ones(num_samps)
    X[:,1:k+1] = np.sin(tk_tiling)
    X[:,k+1:] = np.cos(tk_tiling)
    return X
```

In [9]:
```
def pseudo_fourier(omega, t, A0, a, b):
    assert len(a) == len(b), f"Length of a and length of b must be the s
ame"
    K = len(a)
    s,c = np.zeros(len(t)), np.zeros(len(t))
    for k in range(1, K+1):
        s += a[k-1]*np.sin(k*omega*t)
        c += b[k-1]*np.cos(k*omega*t)
    return A0 + s + c
```

In [99]:
```
def get_prediction(Gband, max_freq, min_freq, k, p=False):

    flux,time=Gband['flux'], Gband['time']
    source_id=Gband['source_id'][0]
    period=estimate_period(Gband, max_freq=max_freq, min_freq=min_freq)
    omega=2*np.pi/period

    X = setup_fit(flux, time, omega, k)
    beta = lstsq(X, flux.data)[0]

    A0=beta[0]
    a=beta[1:k+1]
    b=beta[k+1:]

    time_interp=np.arange(np.min(time), np.max(time), 1)
    f_interp = pseudo_fourier(omega=2*np.pi/period, t=time_interp, A0=A0
, a=a, b=b)
    phase_interp, flux_interp = sort_by_phase(time_interp%period, f_inte
rp)

    if p:
        fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2,2, figsize=(16,7
), gridspec_kw={'height_ratios':[3,1]})
        ax1.scatter(time%period, flux, label="Original",color='black')
        ax1.scatter(time%period, X.dot(beta), label="Predicted", color
='green')
        ax1.set(title=f"SourceID: {source_id}\nRMSE: {rmse(flux, X.dot(b
eta)):.5f}",ylabel="Flux")
        ax1.legend()
        ax1.grid()

        ax3.scatter(time%period, flux-X.dot(beta), label="Residual",colo
r='red')
        ax3.set(xlabel="Phase")
        ax3.grid()

        ax2.scatter(time%period, flux, label="Original f")
        ax2.plot(phase_interp, flux_interp, label="Predicted f", color=
"red")
        ax2.set(title="Interpolated Values", xlabel="Phase",ylabel="Flu
x")
        ax2.legend()
        ax2.grid()

        fig.delaxes(ax4)

        plt.show()
```

In [10]:
```
def sort_by_phase(phase, x):
    sorted_phase, sorted_x = np.array(sorted(zip(phase, x), key=lambda p
air: pair[0])).T
    return sorted_phase, sorted_x
```

In [11]:
```
pf_cond = "pf IS NOT NULL"
rrlyrae_100 = get_gaia_query_rrlyrae_cached(num_stars = 100, num_clean_e
pochs=40, conds=pf_cond)
rrlyrae_100[:10]
```

```
INFO: Query finished. [astroquery.utils.tap.core]
```

Out[11]: *Table length=10*

solution_id	designation	random_index	ref_epoch	ra	
			yr	deg	
int64	object	int64	float64	float64	
1635721458409799680	Gaia DR2 5866125710834119808	841033097	2015.5	212.93756378519396	1.81
1635721458409799680	Gaia DR2 5978435871487788288	1394969254	2015.5	256.52946118354015	0.207
1635721458409799680	Gaia DR2 5704736782734774528	122877659	2015.5	132.81229544277778	0.148
1635721458409799680	Gaia DR2 5816755332315333888	259791940	2015.5	254.94654730343584	0.055
1635721458409799680	Gaia DR2 5821611776409134976	299065082	2015.5	246.3262948830741	0.0259
1635721458409799680	Gaia DR2 5642603243216872576	1311594757	2015.5	132.9564341215911	0.047
1635721458409799680	Gaia DR2 5813181197970338560	1546661016	2015.5	263.35817148226175	0.011
1635721458409799680	Gaia DR2 5630421856972980224	923739124	2015.5	140.99364763000955	0.0560
1635721458409799680	Gaia DR2 5810405553887250432	191264318	2015.5	268.50110356278077	0.0131
1635721458409799680	Gaia DR2 5821156028840408576	1453327615	2015.5	244.35278635111487	0.0163

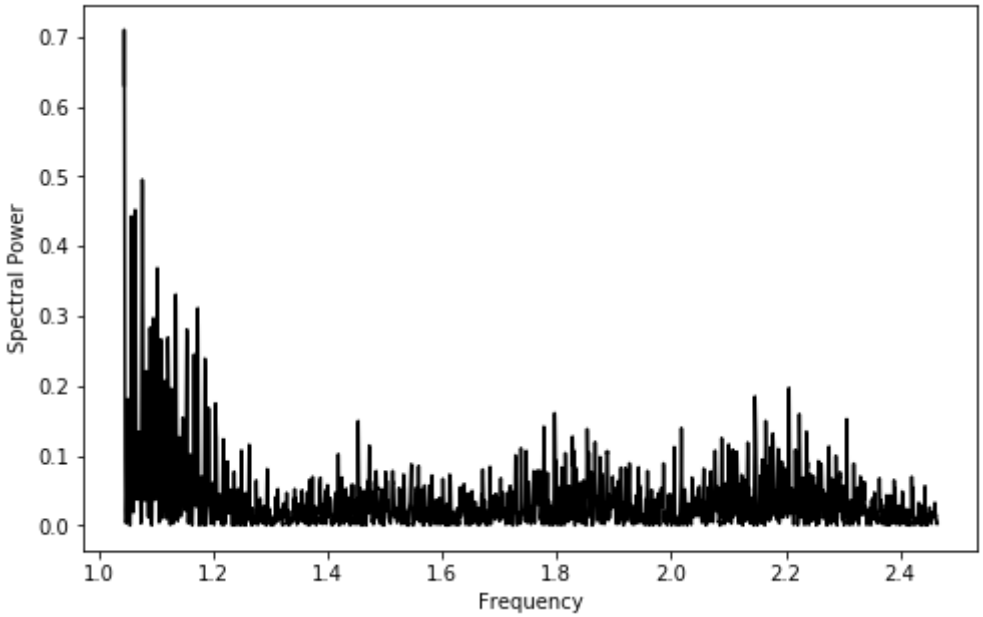
In [22]:
```
for row in rrlyrae_100:
    url = row['epoch_photometry_url']
    source_id = row['source_id']
    dest = f"lightcurve_xmls/{source_id}.xml"
    if os.path.exists(dest):
#         print("It exists!")
        continue
    votable = parse(url.decode('utf-8'))
    votable.format = 'binary'
    with open(dest, 'w') as d:
        votable.to_xml(d)
```

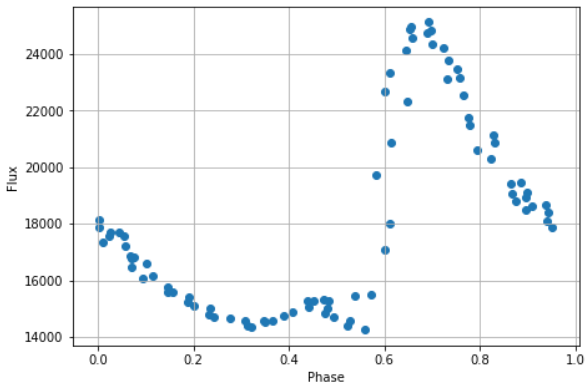
5813181197970338560

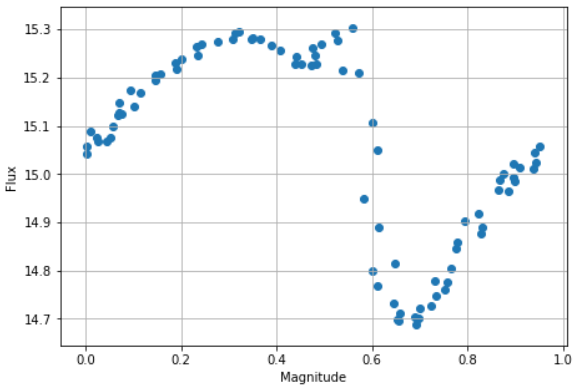
In [47]:
```
selected_index = 6
selected_Gband = get_Gband(rrlyrae_100, selected_index)
```

```
Analyzing star with source_id: 5813181197970338560
```

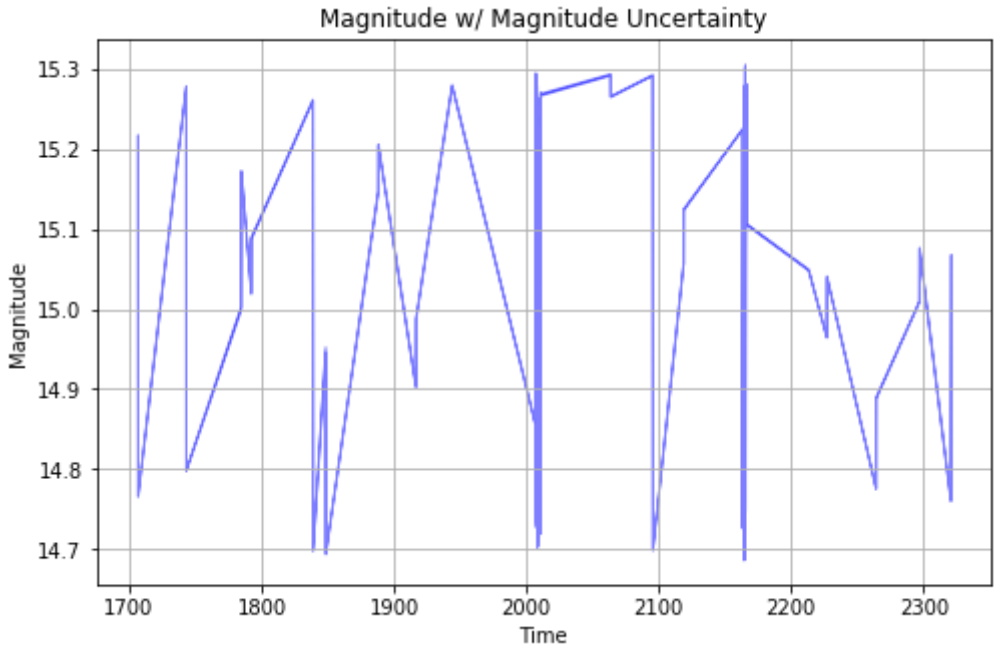
In [48]:
```
period = estimate_period(selected_Gband, p=True)
recorded_period = rrlyrae_100['pf'][selected_index]
print(f"Period as reported by vari_rrlyrae: {recorded_period:.5f}")
print(f"RMSE: {np.sqrt(np.mean(np.square(period - recorded_period)))}")
plot_magnitude(selected_Gband)
```







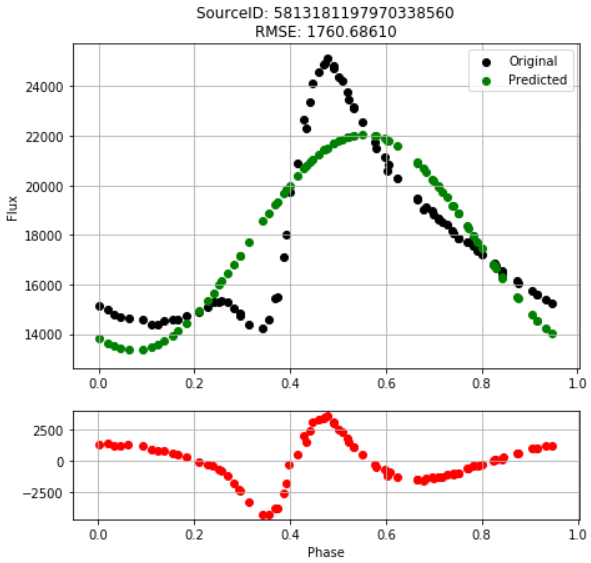

```
Estimated period: 1/1.04433 = 0.95756
Period as reported by vari_rrlyrae: 0.95765
RMSE: 9.488197693585665e-05
```

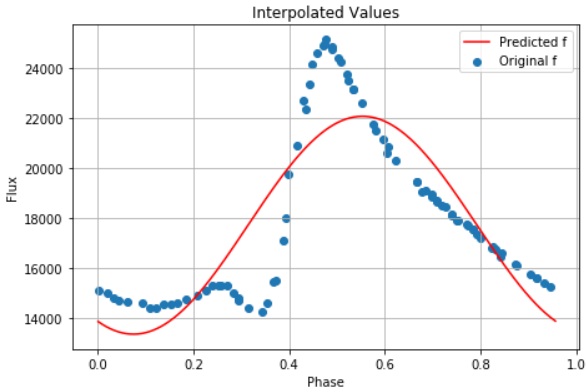



```
Estimated mean: 15.070074092017139
```

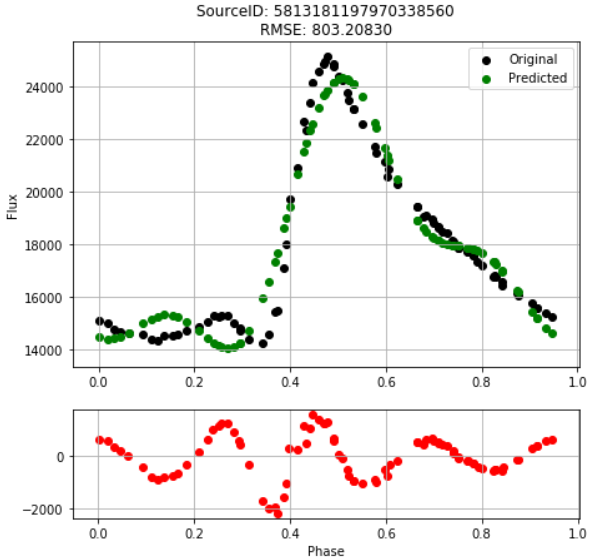
In [108]:
```
k_vals = [1,3,5,7,9]
max_freq, min_freq = get_maxmin_freqs(rrlyrae_100)
for k in k_vals:
    print(f"Analyzing fit for k={k}...")

    get_prediction(selected_Gband, max_freq=max_freq, min_freq=min_freq,
k=k, p=True)
```

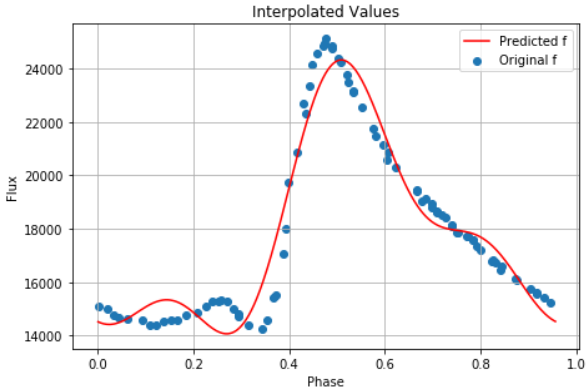
```
Analyzing fit for k=1...
```



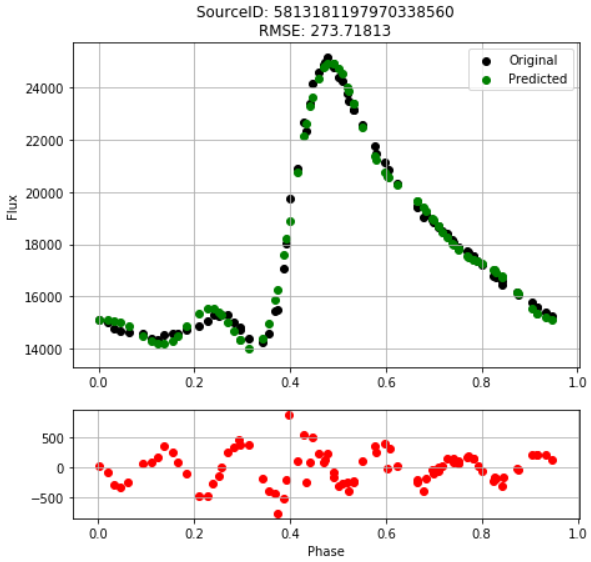


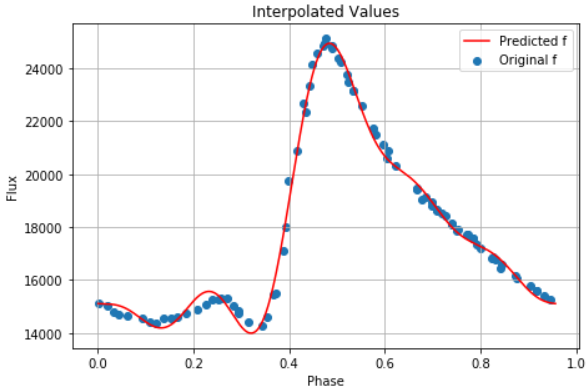

```
Analyzing fit for k=3...
```



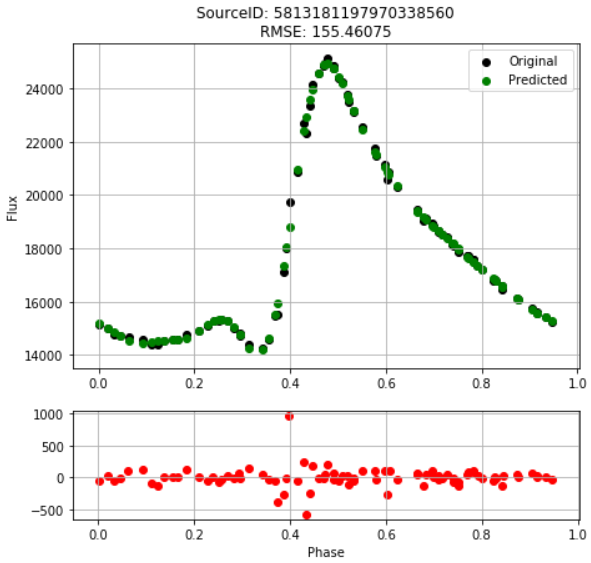


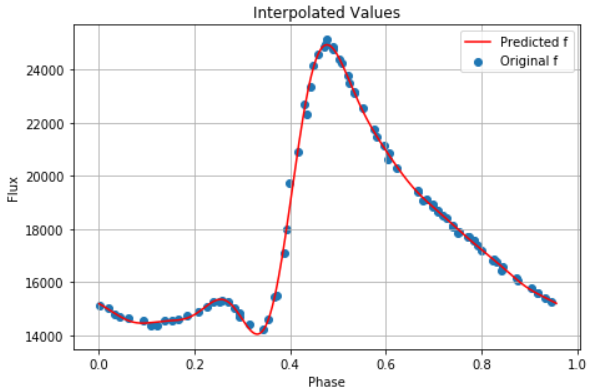

```
Analyzing fit for k=5...
```



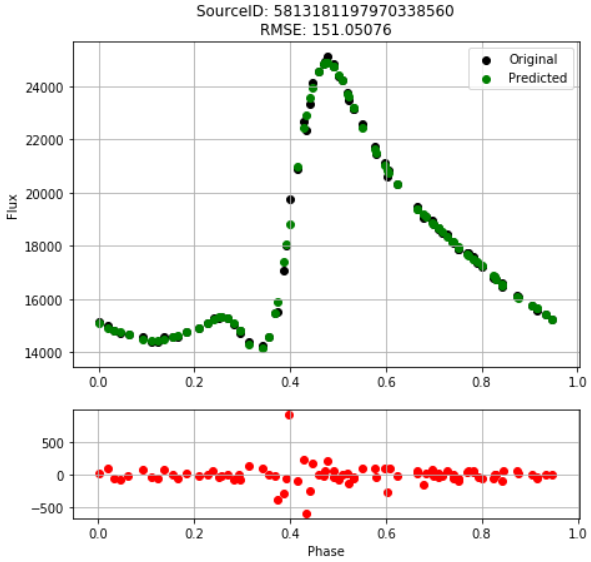


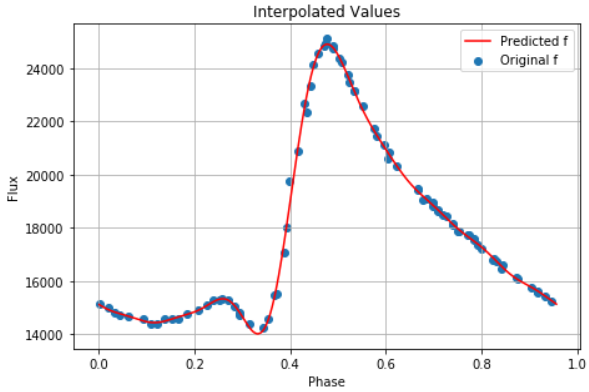
```
Analyzing fit for k=7...
```





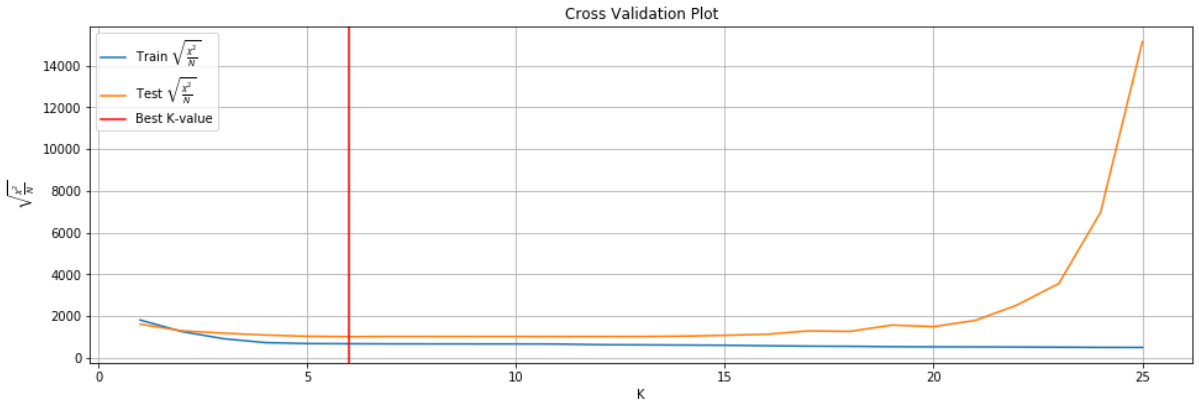

```
Analyzing fit for k=9...
```





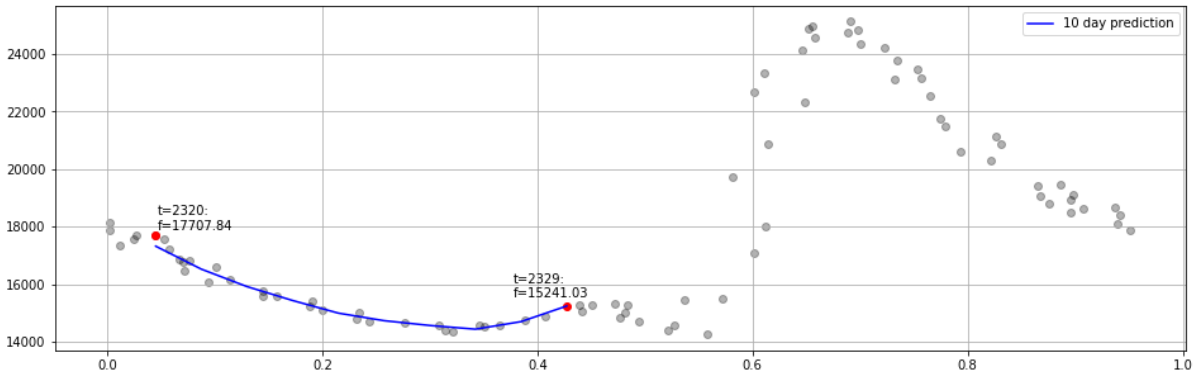
In [56]:
```
k_vals = np.arange(1,25+1)
train_mses = np.zeros(len(k_vals))
test_mses = np.zeros(len(k_vals))
time_train, time_test, flux_train, flux_test = train_test_split(time, fl
ux, test_size=0.2)
```

In [57]:
```
p = False
for i,k in enumerate(k_vals):
#     print(f"Analyzing fit for k={k}...")

    X = setup_fit(flux_train, time_train, omega, k)
    beta_train = lstsq(X, flux_train.data)[0]
    A0=beta_train[0]
    a=beta_train[1:k+1]
    b=beta_train[k+1:]
    pred_train = pseudo_fourier(omega, time_train, A0, a, b)
    pred_test = pseudo_fourier(omega, time_test, A0, a, b)

    train_mse = rmse(flux_train, pred_train)
    train_mses[i] = train_mse

    test_mse = rmse(flux_test, pred_test)
    test_mses[i] = test_mse

    if p and k >= 20:
        fig, (ax1, ax2) = plt.subplots(1,2, figsize=(16,5))
        ax1.scatter(time_train%period,flux_train, label="Truth")
        ax1.scatter(time_train%period,pred_train,label="Pred")
        ax1.set(title=f"K={k}: $\\frac{{\\chi^2}}{{N}} = {train_mse:.3f}
$")
        ax1.legend()

        ax2.scatter(time_test%period,flux_test, label="Truth")
        ax2.scatter(time_test%period,pred_test,label="Pred")
        ax2.set(title=f"$\\frac{{\\chi^2}}{{N}} = {test_mse:.3f}$")
        ax2.legend()
        plt.show()

# print(test_mses)
fig,ax = plt.subplots(figsize=(16,5))
ax.plot(k_vals, train_mses, label="Train $\\sqrt{{\\frac{{\\chi^2}}
{{N}}}}$")
ax.plot(k_vals, test_mses, label="Test $\\sqrt{{\\frac{{\\chi^2}}{{N}}}}
$")
ax.axvline(np.argmin(test_mses)+1,color='red', label="Best K-value")
ax.legend()
ax.grid()
ax.set(xlabel="K", ylabel="$\\sqrt{{\\frac{{\\chi^2}}{{N}}}}$", title="C
ross Validation Plot")
plt.show()
```



In [58]:
```
k_best = np.argmin(test_mses)+1
X = setup_fit(flux, time, omega, k_best)
beta_best = lstsq(X, flux.data)[0]
A0_best=beta_best[0]
a_best=beta_best[1:k_best+1]
b_best=beta_best[k_best+1:]
last_flux, last_time = flux[-1], time[-1]
print(last_flux, last_time)
final_time = last_time+10
time_extrap = np.arange(last_time, final_time)
pred_extrap = pseudo_fourier(omega, time_extrap, A0_best, a_best, b_best
)
phase_extrap, flux_extrap = sort_by_phase(time_extrap%period, pred_extra
p)
fig,ax = plt.subplots(figsize=(16,5))
ax.scatter(time%period, flux,color='black', alpha=0.3)
ax.scatter(last_time%period, last_flux,color='red')
ax.annotate(f"t={last_time:.0f}:\nf={last_flux:.2f}", (last_time%period+
0.002, last_flux+200))
ax.plot(phase_extrap, flux_extrap, color='blue', label="10 day predictio
n")
ax.scatter(phase_extrap[-1], flux_extrap[-1],color='red')
ax.annotate(f"t={time_extrap[-1]:.0f}:\nf={flux_extrap[-1]:.2f}", (phase
_extrap[-1]-0.05, flux_extrap[-1]+300))
ax.grid()
ax.legend()
plt.show()
```

```
17707.83890462764 2320.202939518489
```



In [110]:
```
rrc_conds = f"""
best_classification='RRc'
AND int_average_g > 15
"""
rrlyrae_rrc = get_gaia_query_rrlyrae_cached(num_stars = 3, num_clean_epochs=80, conds=rrc_conds)
rrlyrae_rrc
```

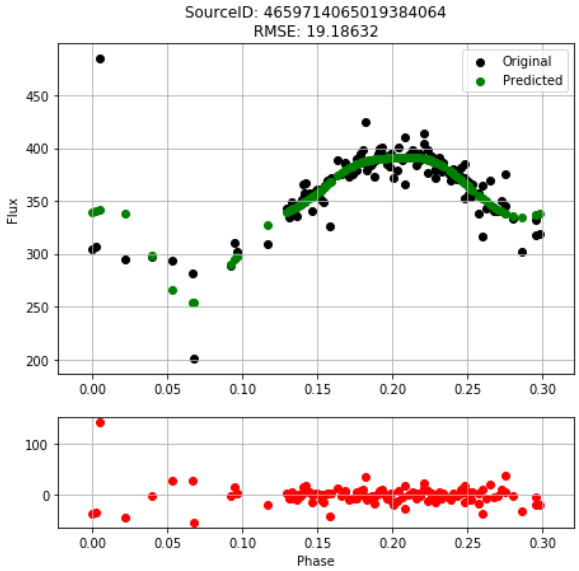
Out[110]: *Table length=3*

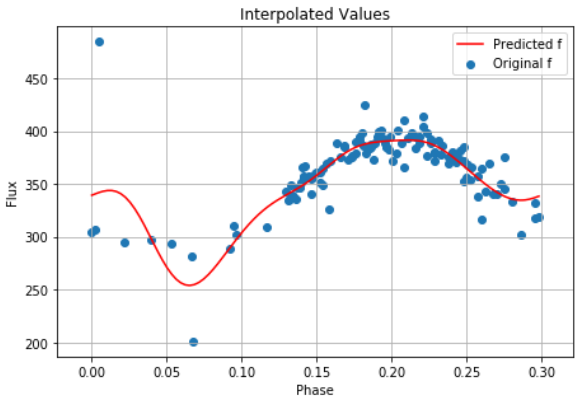
solution_id	designation	random_index	ref_epoch	ra	
			yr	deg	
int64	object	int64	float64	float64	
1635721458409799680	Gaia DR2 4659714065019384064	1255299599	2015.5	89.71078260193757	0.3040
1635721458409799680	Gaia DR2 413458020354821504	1251053435	2015.5	19.111954139381382	0.0238
1635721458409799680	Gaia DR2 5438044842945235712	865958970	2015.5	142.96098504851844	0.0451

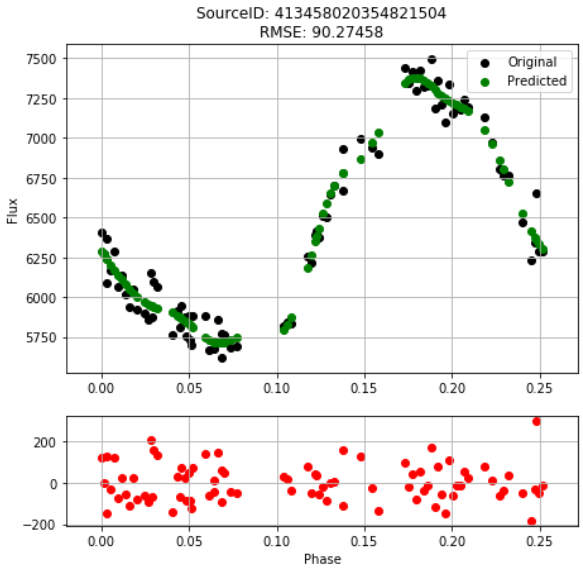
In [111]:
```
rrc_Gbands = [get_Gband(rrlyrae_rrc, i) for i, row in enumerate(rrlyrae_rrc)]
```

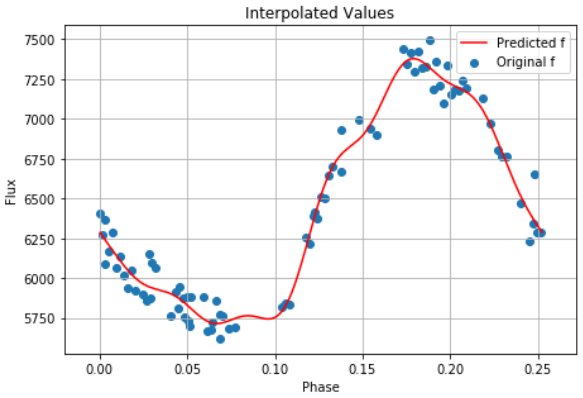
```
Analyzing star with source_id: 4659714065019384064
Analyzing star with source_id: 413458020354821504
Analyzing star with source_id: 5438044842945235712
```

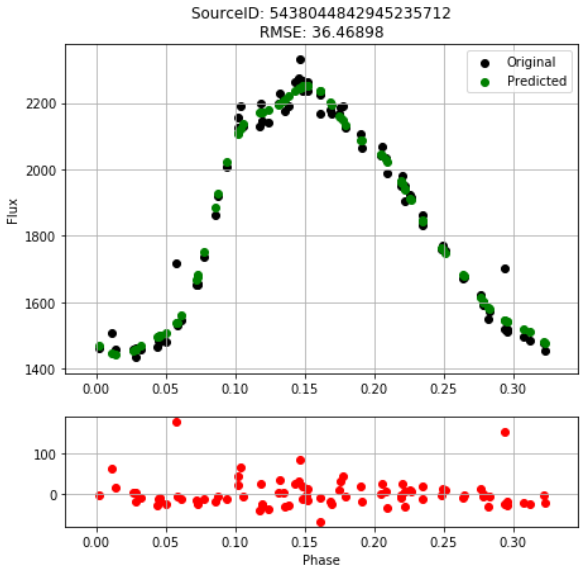
In [112]:
```
max_freq, min_freq = get_maxmin_freqs(rrlyrae_rrc)
for Gband in rrc_Gbands:
    get_prediction(Gband,max_freq=max_freq, min_freq=min_freq, k=k_best,
p=True)
```

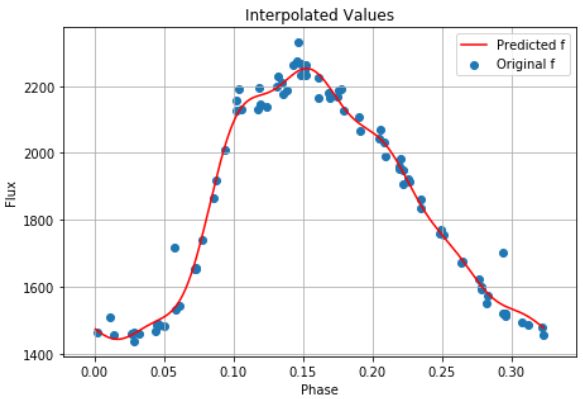

SourceID: 4659714065019384064
RMSE: 19.18632
Original
Predicted
Flux
Phase


Interpolated Values
Predicted f
Original f
Flux
Phase


SourceID: 413458020354821504
RMSE: 90.27458
Original
Predicted
Flux
Phase


Interpolated Values
Predicted f
Original f
Flux
Phase


SourceID: 5438044842945235712
RMSE: 36.46898
Original
Predicted
Flux
Phase


Interpolated Values
Predicted f
Original f
Flux
Phase

In [113]:
```
rrab_conds = f"""
best_classification='RRab'
AND int_average_g > 15
"""
rrlyrae_rrab = get_gaia_query_rrlyrae_cached(num_stars = 3, num_clean_ep
ochs=80, conds=rrab_conds)
rrlyrae_rrab
```

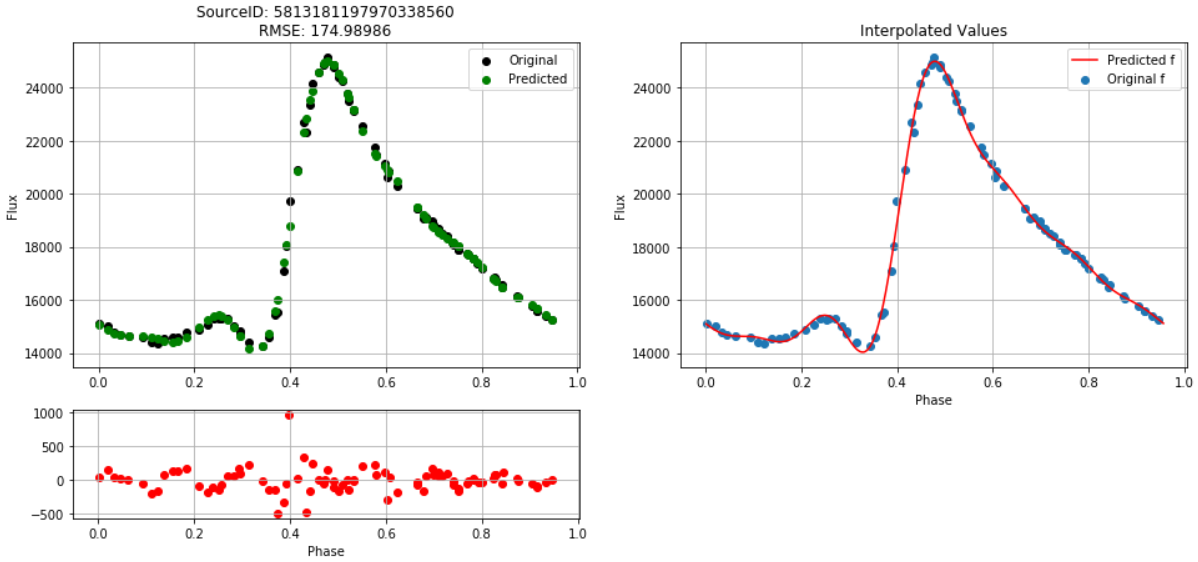
Out[113]: *Table length=3*

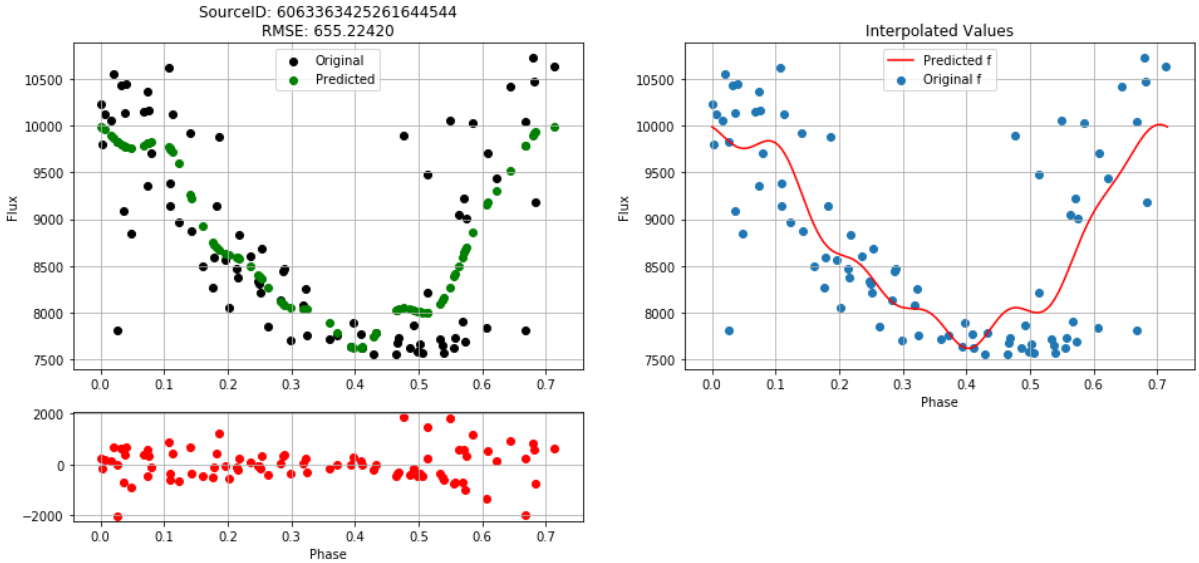
solution_id	designation	random_index	ref_epoch	ra	
			yr	deg	
int64	object	int64	float64	float64	
1635721458409799680	Gaia DR2 5813181197970338560	1546661016	2015.5	263.35817148226175	0.011
1635721458409799680	Gaia DR2 6063363425261644544	307736288	2015.5	197.52702038202472	0.0206
1635721458409799680	Gaia DR2 4180530812989384 96	516077856	2015.5	8.774802583044732	0.114

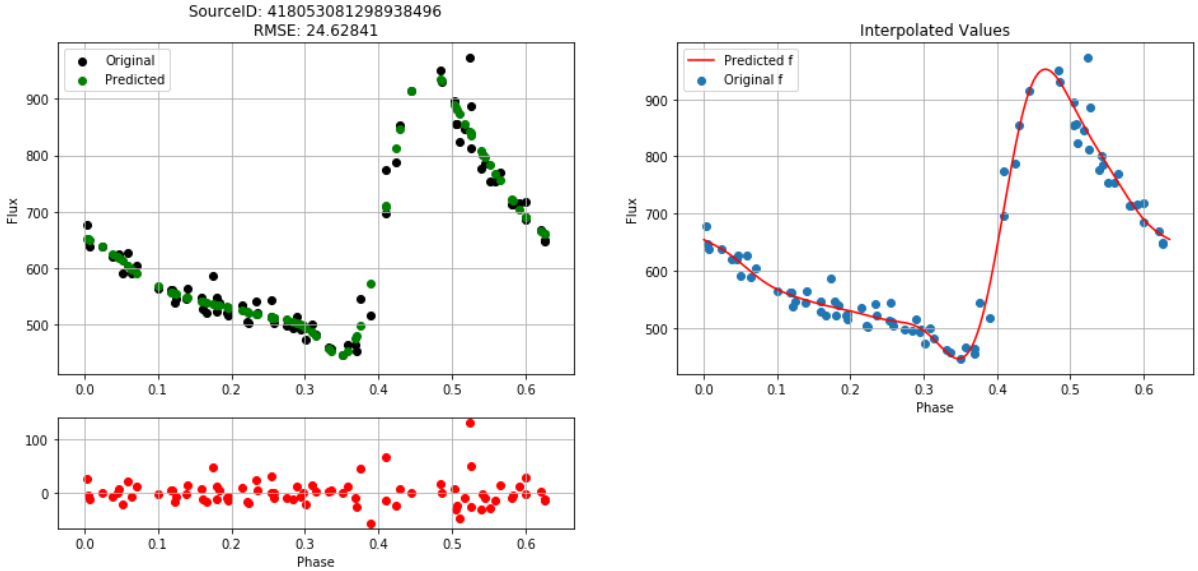
In [114]:
```
rrab_Gbands = [get_Gband(rrlyrae_rrab, i) for i, row in enumerate(rrlyra
e_rrab)]
```

```
Analyzing star with source_id: 5813181197970338560
Analyzing star with source_id: 6063363425261644544
Analyzing star with source_id: 4180530812989384 96
```

In [116]:
```
max_freq, min_freq = get_maxmin_freqs(rrlyrae_rrab)
for Gband in rrab_Gbands:
    get_prediction(Gband, max_freq=max_freq, min_freq=min_freq, k=k_best
, p=True)
```







In [151]:
```
query="""
SELECT *
FROM gaiaedr3.gaia_source as gaia3
JOIN gaiaedr3.dr2_neighbourhood as gaia2
    ON gaia2.dr3_source_id = gaia3.source_id
JOIN gaiadr2.vari_rrlyrae as rrlyrae
    ON gaia2.dr2_source_id = rrlyrae.source_id
JOIN external.gaiaedr3_distance as dists
    ON gaia3.source_id = dists.source_id
WHERE
    abs(b)>30
    AND parallax>0.25
    AND parallax_over_error>5
    AND pf IS NOT NULL
"""
rrlyrae_gaia3 = get_gaia_query_general_cached(query)
print(f"Got {len(rrlyrae_gaia3)} stars")
rrlyrae_gaia3[:10]
```

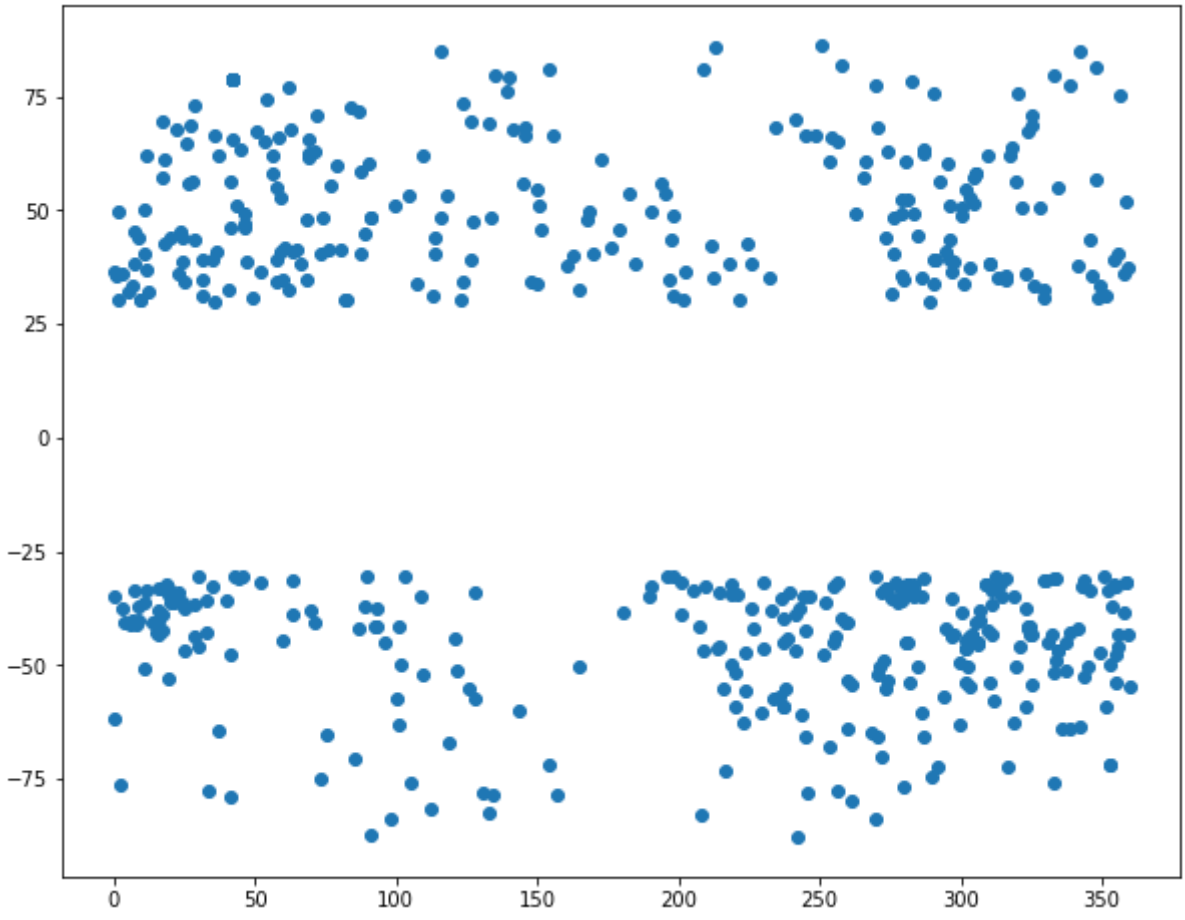
```
INFO: Query finished. [astroquery.utils.tap.core]
Got 548 stars
```

Out[151]: *Table length=10*

solution_id	designation	source_id	random_index	ref_epoch	
				yr	
int64	object	int64	int64	float64	
1636042515805110273	Gaia EDR3 4657904848780392832	4657904848780392832	1297321964	2016.0	82.1
1636042515805110273	Gaia EDR3 4798586678570579072	4798586678570579072	279121884	2016.0	83.4
1636042515805110273	Gaia EDR3 4685757887726594816	4685757887726594816	1480418546	2016.0	11.23
1636042515805110273	Gaia EDR3 4689637956899105792	4689637956899105792	1084076130	2016.0	5.919
1636042515805110273	Gaia EDR3 4689637961175354240	4689637961175354240	47042557	2016.0	5.91
1636042515805110273	Gaia EDR3 4700326863447344768	4700326863447344768	267955142	2016.0	39.2
1636042515805110273	Gaia EDR3 4690886594062571008	4690886594062571008	1401140051	2016.0	15.77
1636042515805110273	Gaia EDR3 4658012051209225984	4658012051209225984	847616141	2016.0	81.9
1636042515805110273	Gaia EDR3 4658145092114070144	4658145092114070144	1681799523	2016.0	79.6
1636042515805110273	Gaia EDR3 4685834544315524224	4685834544315524224	398366609	2016.0	11.38

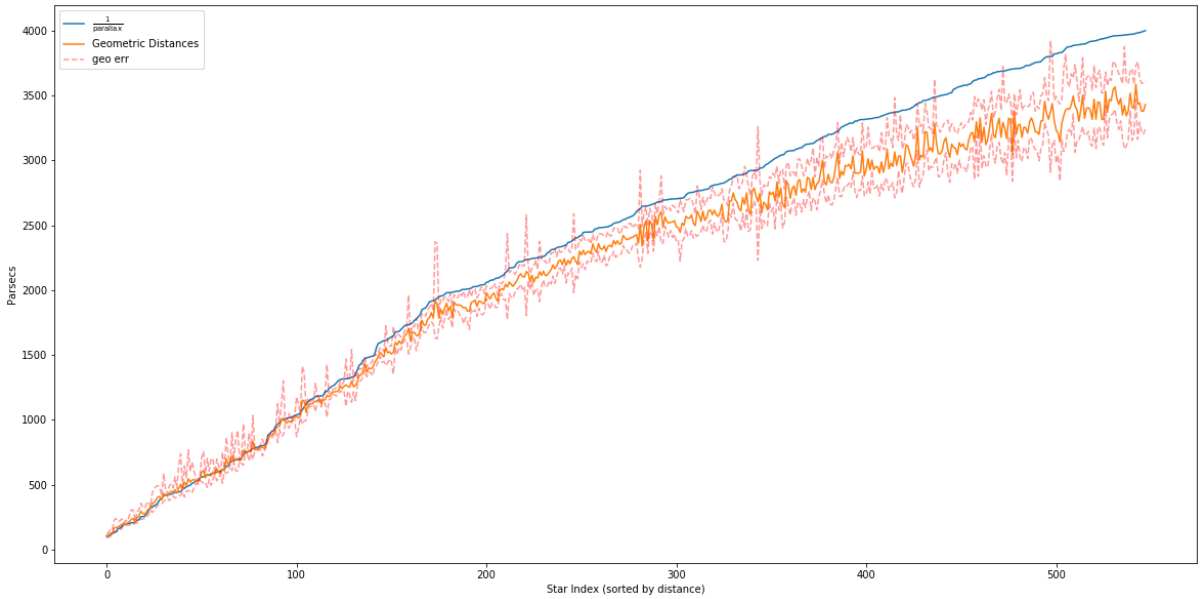
In [152]:
```
plt.figure(figsize=(10,8))
plt.scatter(rrlyrae_gaia3['l'], rrlyrae_gaia3['b'])
```

Out[152]: <matplotlib.collections.PathCollection at 0x1a26c48250>



In [159]:
```
rrlyrae_gaia3_df = rrlyrae_gaia3.to_pandas()
dists = pd.concat(
        [rrlyrae_gaia3_df[["r_med_geo", "r_hi_geo", "r_lo_geo"]], 1000/r
rlyrae_gaia3_df['parallax']],
        axis=1
    ).rename(columns={"parallax":"distance"}).sort_values("distance")
n = np.arange(dists.shape[0])
plt.figure(figsize=(20,10))
plt.plot(n,dists["distance"], label=r"$\frac{1}{\mathrm{parallax}}$")
plt.plot(n, dists["r_med_geo"], label="Geometric Distances")
plt.plot(n, dists["r_hi_geo"], "--", color="red", alpha=0.4, label="geo
 err")
plt.plot(n, dists["r_lo_geo"], "--", color="red", alpha=0.4)
plt.ylabel("Parsecs")
plt.xlabel("Star Index (sorted by distance)")
plt.legend()
plt.show()
```

```
/Users/Rafferino/anaconda3/lib/python3.7/site-packages/IPython/core/int
eractiveshell.py:3331: TableReplaceWarning: converted column 'phot_bp_n
_contaminated_transits' from integer to float
  exec(code_obj, self.user_global_ns, self.user_ns)
/Users/Rafferino/anaconda3/lib/python3.7/site-packages/IPython/core/int
eractiveshell.py:3331: TableReplaceWarning: converted column 'phot_bp_n
_blended_transits' from integer to float
  exec(code_obj, self.user_global_ns, self.user_ns)
/Users/Rafferino/anaconda3/lib/python3.7/site-packages/IPython/core/int
eractiveshell.py:3331: TableReplaceWarning: converted column 'phot_rp_n
_contaminated_transits' from integer to float
  exec(code_obj, self.user_global_ns, self.user_ns)
/Users/Rafferino/anaconda3/lib/python3.7/site-packages/IPython/core/int
eractiveshell.py:3331: TableReplaceWarning: converted column 'phot_rp_n
_blended_transits' from integer to float
  exec(code_obj, self.user_global_ns, self.user_ns)
```



In [160]:
```
query = """
    SELECT *
    FROM gaiadr2.gaia_source as gaia2
    JOIN gaiaedr3.dr2_neighbourhood as link
        ON gaia2.source_id = link.dr2_source_id
    JOIN gaiadr2.vari_rrlyrae as rrlyrae
        ON rrlyrae.source_id = gaia2.source_id
    WHERE
        parallax_over_error > 5
        AND abs(b) > 30
        AND parallax > 0.25
        AND pf IS NOT NULL
"""
rrlyrae_gaia2 = get_gaia_query_general_cached(query)
print(f"Got {len(rrlyrae_gaia2)} stars")
rrlyrae_gaia2[:10]
```

```
INFO: Query finished. [astroquery.utils.tap.core]
Got 518 stars
```

Out[160]: *Table length=10*

solution_id	designation	source_id	random_index	ref_epoch	
				yr	
int64	object	int64	int64	float64	
1635721458409799680	Gaia DR2 4654631533876789632	4654631533876789632	502718668	2015.5	74
1635721458409799680	Gaia DR2 4654631533876789632	4654631533876789632	502718668	2015.5	74
1635721458409799680	Gaia DR2 4685757887726594816	4685757887726594816	812747682	2015.5	11.2
1635721458409799680	Gaia DR2 4689637956899105792	4689637956899105792	1530397305	2015.5	5.9
1635721458409799680	Gaia DR2 4689637956899105792	4689637956899105792	1530397305	2015.5	5.9
1635721458409799680	Gaia DR2 4700326863447344768	4700326863447344768	642465858	2015.5	39.2
1635721458409799680	Gaia DR2 4658186602866639872	4658186602866639872	1491505400	2015.5	79.0
1635721458409799680	Gaia DR2 4658186602866639872	4658186602866639872	1491505400	2015.5	79.0
1635721458409799680	Gaia DR2 4632502526615290624	4632502526615290624	1511122055	2015.5	32.43
1635721458409799680	Gaia DR2 4632502526615290624	4632502526615290624	1511122055	2015.5	32.43

In [173]:
```
mask = [(source_id in rrlyrae_gaia2["dr3_source_id"]) for source_id in r
rlyrae_gaia3["source_id"]]
gaia3_data = rrlyrae_gaia3[mask].to_pandas().sort_values("parallax")
gaia3_data["dr3_source_id"]
```

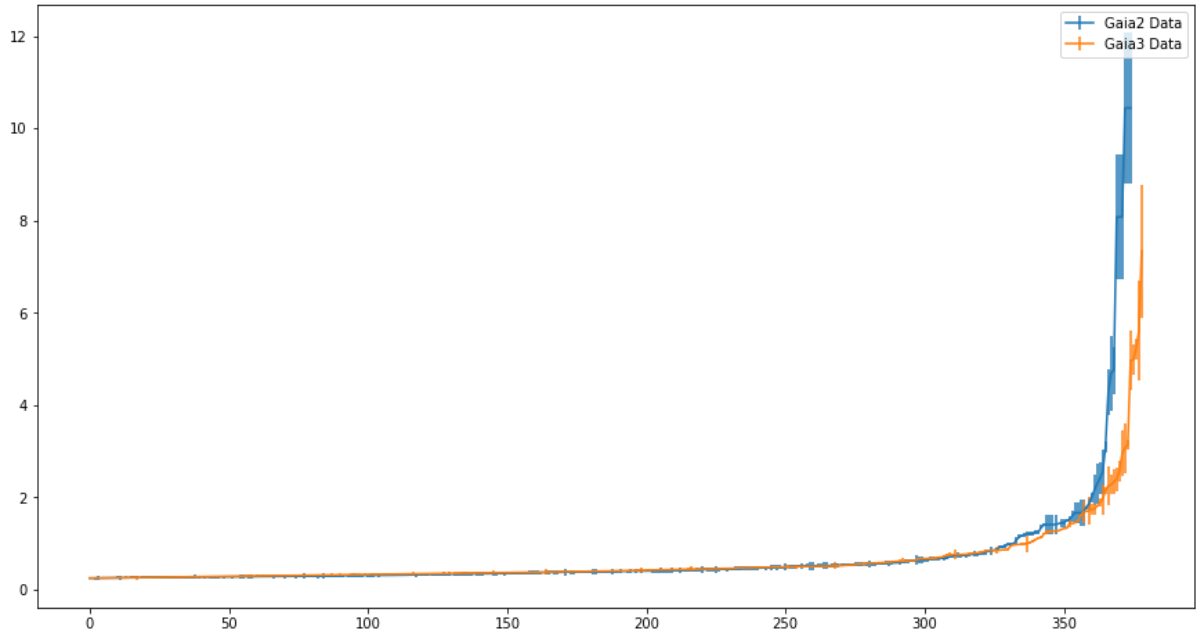
Out[173]:
```
271    3186033648144893696
340    6910756005950942336
93     3945610468551775616
171     612591803903942016
183    1540009731422069504
               ...
61     4943063090574101376
63     4955988365155969792
3      4700326863447344768
279    4998160202357316096
179    1328057321618952064
Name: dr3_source_id, Length: 379, dtype: int64
```

In [176]:
```
mask2 = [(source_id in rrlyrae_gaia3["dr3_source_id"]) for source_id in
rrlyrae_gaia2["source_id"]]
gaia2_data = rrlyrae_gaia2[mask2].to_pandas().sort_values("parallax")
len(gaia2_data)
```

```
/Users/Rafferino/anaconda3/lib/python3.7/site-packages/IPython/core/int
eractiveshell.py:3331: TableReplaceWarning: converted column 'priam_fla
gs' from integer to float
  exec(code_obj, self.user_global_ns, self.user_ns)
/Users/Rafferino/anaconda3/lib/python3.7/site-packages/IPython/core/int
eractiveshell.py:3331: TableReplaceWarning: converted column 'flame_fla
gs' from integer to float
  exec(code_obj, self.user_global_ns, self.user_ns)
```

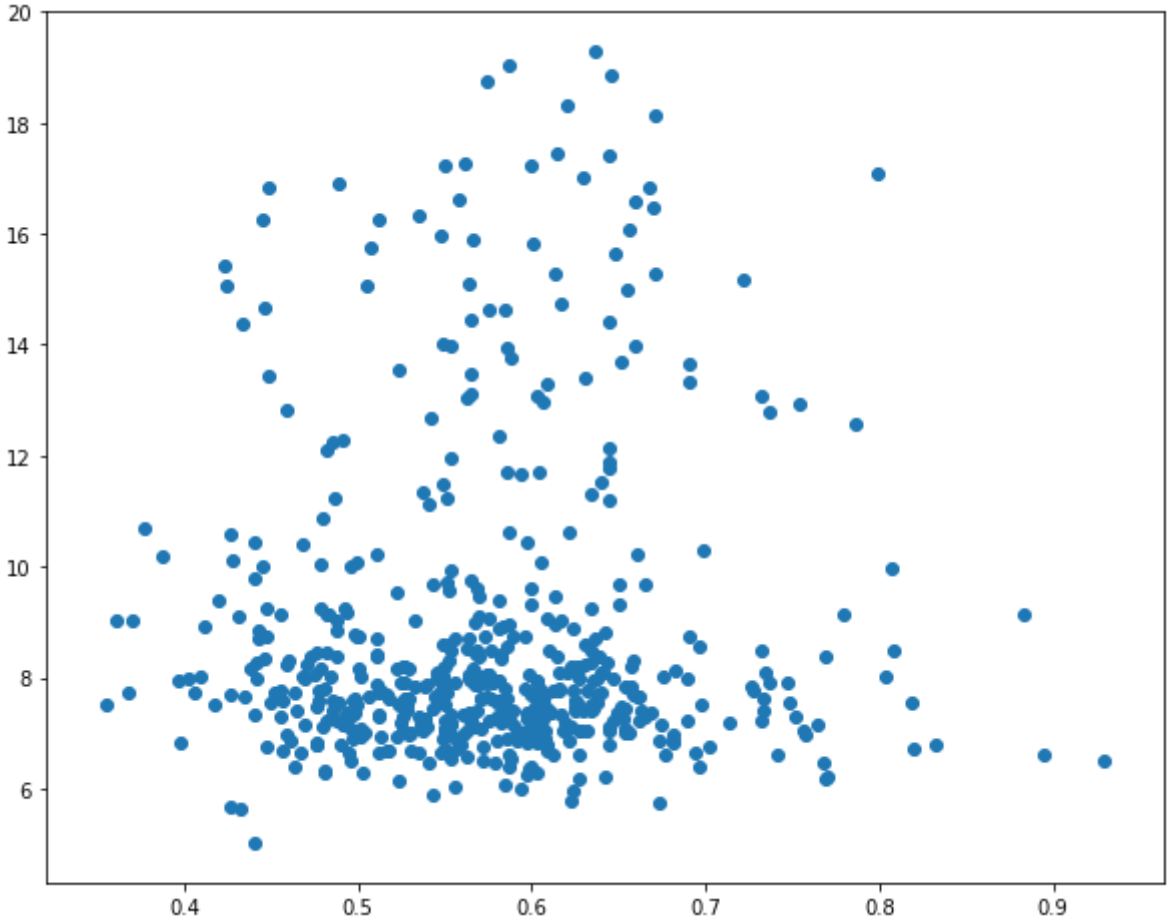
Out[176]:
```
375
```

In [177]:
```
plt.figure(figsize=(15,8))
plt.errorbar(np.arange(len(gaia2_data)), gaia2_data["parallax"], yerr=ga
ia2_data["parallax_error"], label="Gaia2 Data")
plt.errorbar(np.arange(len(gaia3_data)), gaia3_data["parallax"], yerr=ga
ia3_data["parallax_error"], label="Gaia3 Data")
plt.legend()
plt.show()
```



In [179]:
```
Mg = rrlyrae_gaia3["phot_g_mean_mag"]-5*np.log10((rrlyrae_gaia3["r_hi_geo"]-rrlyrae_gaia3["r_lo_geo"])/2)+5
plt.figure(figsize=(10,8))
plt.scatter(rrlyrae_gaia3["pf"], Mg)
```

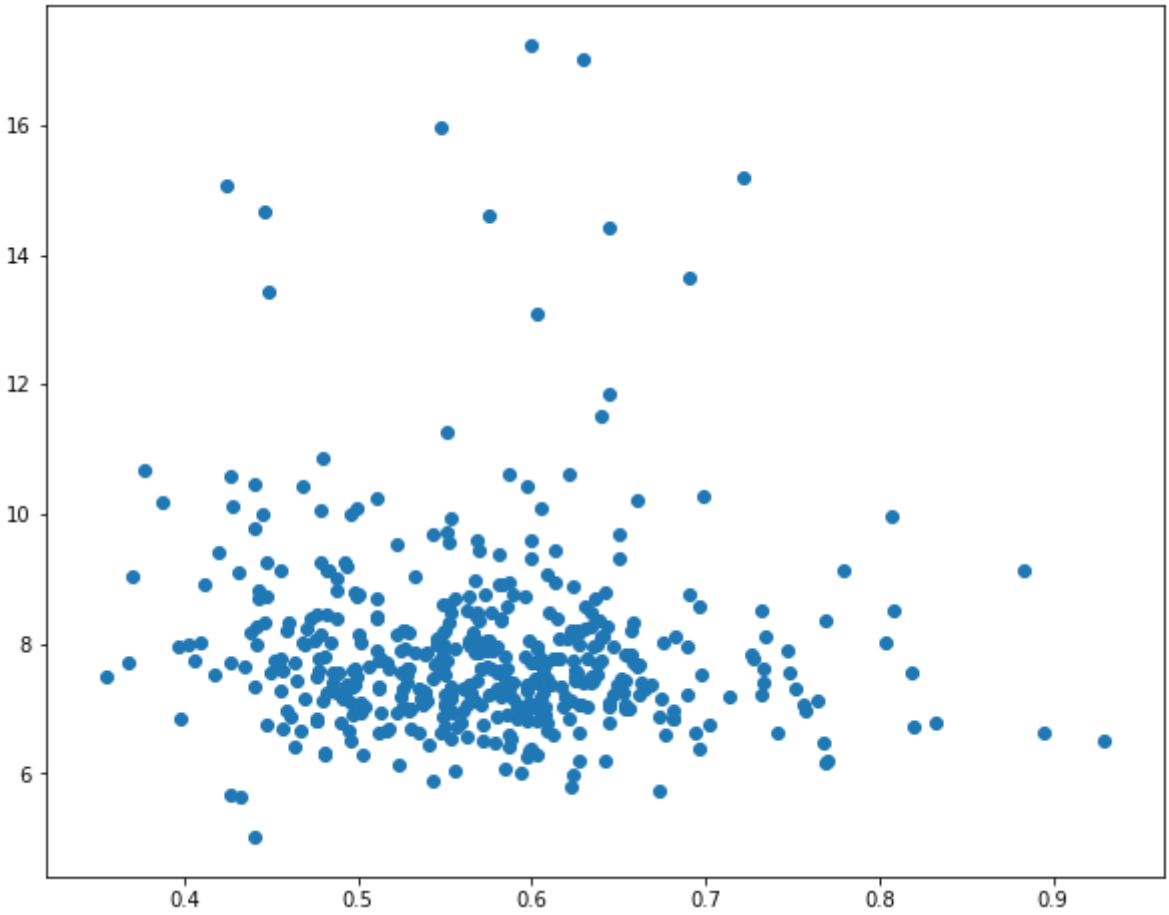
Out[179]: <matplotlib.collections.PathCollection at 0x1a27edeb50>



In [ ]:
```
query = """
    SELECT *
    FROM gaiaedr3.gaia_source as gaia
    JOIN gaiaedr3.dr2_neighbourhood as link
        ON gaia.source_id = link.dr3_source_id
    JOIN gaiadr2.vari_rrlyrae as lyrae
        ON lyrae.source_id = link.dr2_source_id
    JOIN external.gaiaedr3_distance as dists
        ON gaia.source_id = dists.source_id
    WHERE
        parallax_over_error > 5
        AND abs(b) > 30
        AND parallax > 0.25
        AND pf IS NOT NULL
        AND astrometric_excess_noise < 1
        AND 1 + 0.015*power(bp_rp, 2) < phot_bp_rp_excess_factor
        AND phot_bp_rp_excess_factor < 1.3 + 0.06*power(bp_rp, 2)
"""

rrlyrae_gaia3_filt = get_gaia_query_general_cached(query)
```

In [181]:
```
Mg = rrlyrae_gaia3_filt["phot_g_mean_mag"]-5*np.log10((rrlyrae_gaia3_fil
t["r_hi_geo"]-rrlyrae_gaia3_filt["r_lo_geo"])/2)+5
plt.figure(figsize=(10,8))
plt.scatter(rrlyrae_gaia3_filt["pf"], Mg)
```

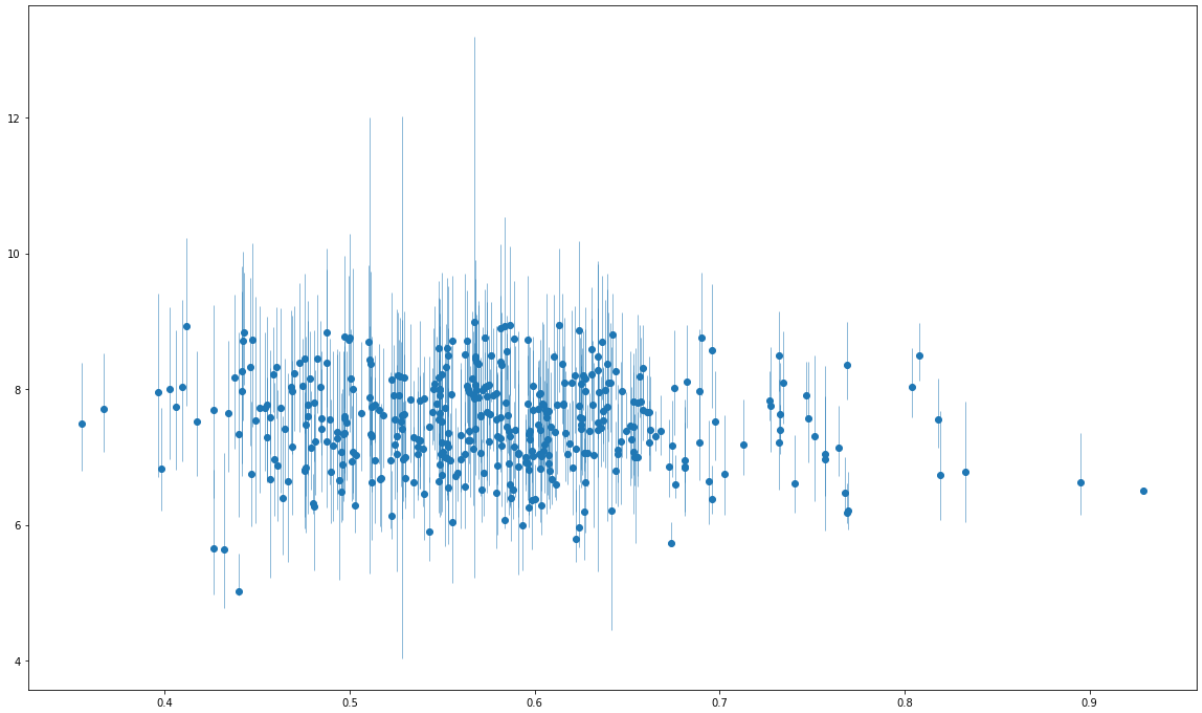
Out[181]: <matplotlib.collections.PathCollection at 0x1a25617d90>



In [183]:
```
rrlyrae_gaia3_filt_df = rrlyrae_gaia3_filt.to_pandas()
Mg = rrlyrae_gaia3_filt_df["phot_g_mean_mag"]-5*np.log10((rrlyrae_gaia3_
filt_df["r_hi_geo"]-rrlyrae_gaia3_filt_df["r_lo_geo"])/2)+5
rrlyrae_gaia3_filt_df = pd.concat([rrlyrae_gaia3_filt_df,
                                   Mg.rename("Mg")],
                                   axis=1)
rrlyrae_gaia3_filt_df = rrlyrae_gaia3_filt_df[rrlyrae_gaia3_filt_df["Mg"
]<9]
```

In [190]:
```
high_error = rrlyrae_gaia3_filt_df["phot_g_mean_mag"]-5*np.log10(rrlyrae
_gaia3_filt_df["r_hi_geo"])+5
low_error = rrlyrae_gaia3_filt_df["phot_g_mean_mag"]-5*np.log10(rrlyrae_
gaia3_filt_df["r_lo_geo"])+5
error = np.stack((high_error, low_error))
```

In [191]:
```
plt.figure(figsize=(20,12))
plt.errorbar(rrlyrae_gaia3_filt_df["pf"], rrlyrae_gaia3_filt_df["Mg"], y
err=error, fmt="o", elinewidth=0.5)
plt.show()
```



In [ ]: